

ERDC/EL TR-02-29

Environmental Laboratory



**US Army Corps
of Engineers®**
Engineer Research and
Development Center

User's Manual for NAVEFF Navigation Effects Model

Scott Bourne

September 2002

20021129 075

The contents of this report are not to be used for advertising, publication, or promotional purposes. Citation of trade names does not constitute an official endorsement or approval of the use of such commercial products.

The findings of this report are not to be construed as an official Department of the Army position, unless so designated by other authorized documents.



PRINTED ON RECYCLED PAPER

User's Manual for NAVEFF Navigation Effects Model

by Scott Bourne

Environmental Laboratory
U.S. Army Engineer Research and Development Center
3909 Halls Ferry Road
Vicksburg, MS 39180-6199

Final report

Approved for public release; distribution is unlimited

Contents

Preface	iv
1—Introduction	1
System Requirements.....	1
Input File Verification	2
2—Program and Output Verification	3
Verification Method.....	3
Results of Verification	3
3—Program Installation and Execution	4
Appendix A: File Formats	A1
Appendix B: CHECK Program	B1
Appendix C: NAVEFF Program	C1
SF 298	

Preface

This report was prepared for the Upper Mississippi River Navigation Study (UMRS) sponsored by the U.S. Army Engineer District, Rock Island. The user guide was written by Scott Bourne, Environmental Laboratory (EL), U.S. Army Engineer Research and Development Center (ERDC), Vicksburg, MS. Dr. Steve Maynard, Coastal and Hydraulics Laboratory (CHL), ERDC, provided technical review. Project manager for the UMRS was Dr. Kenneth Barr, Rock Island District.

This work was conducted under the direction of Dr. David J. Tazik, Chief, Ecosystem Evaluation and Engineering Division, EL, and Mr. Harold W. West, Chief, Environmental Systems Branch. Director of EL was Dr. Edwin A. Theriot.

At the time of publication of this report, Dr. James R. Houston was Director of ERDC, and COL John W. Morris III, EN, was Commander and Executive Director.

This report should be cited as follows:

Bourne, S. (2002). "User's manual for NAVEFF navigation effects model," ERDC/EL TR-02-29, U.S. Army Engineer Research and Development Center, Vicksburg, MS.

The contents of this report are not to be used for advertising, publication, or promotional purposes. Citation of trade names does not constitute an official endorsement or approval of the use of such commercial products.

1 Introduction

The model for Navigation Effects (NAVEFF) models physical forces of tow boat passage across a river cross section. This is a one-dimensional model containing empirical relations for estimating physical forces.¹

The physical forces that are modeled in NAV EFF will be expressed as:

- a.* Secondary wave height.
- b.* Scour.
- c.* Drawdown.
- d.* Velocity change.
- e.* Shear stress.

Physical forces that are generated by NAV EFF are based on 108 boat types. These 108 boat types are a combination of three boat speeds, three sail line positions, three water stages, direction (upstream or downstream), propeller type (open wheel or kort nozzle), boat size, and draft. The fleet characteristics are described in Appendix A (filename P#_PARM.DAT).

System Requirements

These computer programs have been tested and run on both UNIX and Windows. A comparison of a UNIX run and a Windows run showed that the output was identical. Both the CHECK and NAV EFF programs were compiled on both systems using Visual Fortran.

Due to the size of the output files, a large amount of hard disk space will generally be required. Typically 1 to 3 GB of free disk space will be needed. Pools 4, 8, 13, 26, and Lagrange will require more disk space since the program will be executed at each half river mile.

¹ Maynard, S., Bourne, S., Graves, M., Landwehr, K., and Knight, S., "UMR-IWW System Models Report – Physical Effects Model," U.S. Army Engineer District, Rock Island, Rock Island, IL, in preparation.

Input File Verification

NAVEFF requires six input files. A CHECK program (CHECK.EXE), developed to test these input files, verifies that the input files are in the proper format. The check program looks at each sail line position and ascertains if the cell selected for the sail line and the two adjacent cells are deep enough for a tow with a 9-ft (2.743-m) draft to pass. The program checks the cells that are within the channel extents (extents defined in P#_chan.dat input file, Coastal and Hydraulics Laboratory (CHL), U.S. Army Engineer Research and Development Center, Vicksburg, MS) and determines if each of these cells has ambient velocity values and a sediment type. The output from the CHECK program is written to an ASCII file where the results can be viewed.

2 Program and Output Verification

Verification Method

Verification of the NAVEFF code was done by processing 16 randomly selected traffic configurations in Pool 13 and comparing the NAVEFF output results to an earlier BASIC version of NAVEFF. The NAVEFF output parameters, velocity change, drawdown, scour, shear stress, and secondary wave height, were tabulated and graphically displayed for comparison purposes.

Output verification was done by processing the largest, fastest traffic configuration and the light traffic configuration. Four NAVEFF output parameters (change velocity, drawdown, scour, and shear stress) were selected, and the results were displayed to determine if there was something obviously wrong with the output. The largest, fastest configuration used was the following boat type: upbound direction, fast speed, big boat, loaded barge, kort nozzle, low stage, and middle sailing line. The light boat traffic configuration used was: downbound direction, slow speed, small boat, empty barge, kort nozzle, high stage, and middle sailing line.

Results of Verification

The comparison of the current NAVEFF code and an earlier BASIC version of the code demonstrated that there were no significant differences in the output generated by both versions.

The output generated by the worst traffic configuration and the light traffic configuration was examined by CHL personnel. Navigation effects values that were generated by these two configurations were determined to fall within an acceptable range of values.

3 Program Installation and Execution

The steps below list the files needed in each directory to execute the CHECK and NAVEFF programs.

Installation

1. Set up a new directory (pool#).
2. Copy the NAVEFF and CHECK program executables (.exe) into the new directory.
3. Copy the following input files into the same directory:
p#_chan.dat p#_sail.dat
p#_lmh.dat p#_elev.dat
p#_parm.dat p#_yz.trf
Where: y = stage, z = sailing line

The format of the input files is indicated in Appendix A

Execution of CHECK Program

4. On the command line of a DOS or UNIX window, type CHECK. The CHECK program code is listed in Appendix B.
5. The user will be prompted for the pool number:
c:\pool#\>check
Enter Pool Number: p#
6. When the program is finished, the message NORMAL STOP CONDITIONS will be written to the screen.
7. Two ASCII files will be generated by executing this command, an error file (.err), which will list any errors in the input files, and an output file (.out), which will always be empty.
8. The errors found in the error file refer to the relationship of the p#_sail.dat and p#_chan.dat files to the p#_elev.dat file.
9. Sail line errors are found when the cell selected for the sail line and two adjacent cells have a water depth value of less than 9 ft (2.743 m). Edit the p#_sail.dat file to move the sail line position. The left, middle, and right sail line positions are the same for each water stage. When a sail line position is being moved, be sure and move it for each water stage level.

10. Cells without ambient velocity or sediment type that are within the channel extents set in the p#_chan.dat file will produce an error in the error file. These errors are water stage level specific, meaning that the channel extents for the three water stages are different and that error refers to a specific water stage. This error will list all the cells that do not have an ambient velocity value. Edit the p#_chan.dat and move the channel extent in toward the sail line until all cells have an ambient velocity value.
11. Once all the errors have been corrected, delete the error and output files and re-run the CHECK program.
12. Sometimes moving the channel and sail line will generate more errors, so always re-run the CHECK program after making edits.
13. When the error file comes back empty, the NAVEFF program is ready to be run.

Execution of NAVEFF Program

14. On the command line of a DOS or UNIX window, type NAVEFF. The NAVEFF program code is listed in Appendix C.
15. The user will be prompted for the traffic file:
c:\pool#\>naveff
Enter Traffic File Name: p#_yz.trf
Where y = Stage, z = sail line
16. NAVEFF will be run nine times for each of the nine traffic files per pool.
17. Each line in the output file contains a unique ID that allows the output data to be incorporated back into the geographic information system database and displayed.

Appendix A

File Formats

The NAVEFF program requires six input files and generates a single output file. These input and output files are ASCII files and comma delimited. The formats and descriptions of the input files are listed in this section.

File Name	Description
P#_CHAN.DAT	This input file contains the bank line position at low, medium, and high stage for each river mile.
P#_ELEV.DAT	This input file contains the ambient velocities at low, medium, and high stage. Also, sediment values are contained in this input file.
P#_LMH.DAT	Low, medium, and high river stage values are contained in this input file.
P#_PARM.DAT	This input file contains the tow boat characteristics.
P#_SAIL.DAT	Left, middle, and right sail line positions are contained in this input file.
P#_YZ.TRF	These input files contain the 108 tow boat types for each river mile.
P#_YZ.OUT	NAVEFF output file.

P#_CHAN.DAT

Bank Positions				
Variable	Description	Type	Format	Example
RIVER_MI	River Mile	Numeric	5.1	523.5
LSTG_LBK	Cell ID Location of Left Bank Line at Low Stage	Character	10	405L5235
LSTG_RBK	Cell ID Location of Right Bank Line at Low Stage	Character	10	305L5235
MSTG_LBK	Cell ID Location of Left Bank Line at Medium Stage	Character	10	505L5235
MSTG_RBK	Cell ID Location of Right Bank Line at Medium Stage	Character	10	605L5235
HSTG_LBK	Cell ID Location of Left Bank Line at High Stage	Character	10	905L5235
HSTG_RBK	Cell ID Location of Right Bank Line at High Stage	Character	10	805L5235

Example:

715.0175L7150 145R7150 175L7150 145R7150 175L7150 155R7150
716.0305L7160 145R7160 305L7160 145R7160 305L7160 175R7160
717.0525L7170 105R7170 525L7170 105R7170 525L7170 105R7170

P#_ELEV.DAT

Cell Information Input File				
Variable	Description	Type	Format	Example
RIVER_MI	River Mile	Numeric	5.1	523.5
CELL_ID	Cell ID	Character	10	305L5235
XCOORD	Mid-Point X Coordinate	Numeric	12.1	713255.2
YCOORD	Mid-Point Y Coordinate	Numeric	12.1	4681465
MSL_M	Mean Sea Level Elevation at Flat Pool (M)	Numeric	8.3	179.821
AMBVEL_L	Ambient Current Velocity at Low Stage (M/S)	Numeric	8.3	0.141
AMBVEL_M	Ambient Current Velocity at Medium Stage (M/S)	Numeric	8.3	0.291
AMBVEL_H	Ambient Current Velocity at High Stage (M/S)	Numeric	8.3	0.346
D50_GSZ_MM	D50 Particle Grain Size (MM)	Numeric	8.3	0.081
D50_VEL_CMS	D50 Particle Fall Velocity (CM/S)	Numeric	8.3	0.531
COH_SED	Cohesive Sediment	Numeric	6	2
COH_CLASS	Cohesive Class for Group 2	Numeric	6	2

Example:

```

715.0175L7150 624274.7 4873457.0 194.706 0.074 0.232 0.648 0.654 8.356 3 0
715.0165L7150 624270.9 4873447.5 194.554 0.077 0.244 0.676 0.654 8.356 3 0
715.0155L7150 624267.2 4873438.5 193.091 0.106 0.343 0.912 0.654 8.356 3 0

```

P#_LMH.DAT

Water Stages at Each River Mile				
Variable	Description	Type	Format	Example
RIVER_MI	River Mile	Numeric	5.1	523.5
LSTG_MSL_M	Low Stage Value in MSL (M)	Numeric	8.2	177.49
MSTG_MSL_M	Medium Stage Value in MSL (M)	Numeric	8.2	177.76
HSTG_MSL_M	High Stage Value	Numeric	8.2	177.88

Example:

715.0 196.69 196.47 196.81
716.0 196.69 196.50 196.96
717.0 196.69 196.54 197.11

P#_PARM.DAT

Boat Characteristics			
(Position 1)	(Position 2)	(Position 9)	Remark
#	Speed		
S	2.24		Slow speed (M/S)
M	2.91		Medium speed (M/S)
F	3.58		Fast speed (M/S)
(Blank Row)			
#	Size		
L	10.67	(1 Blank) 45.72	Light size (M)
S	10.67	178.31	Small size (M)
M	21.34	237.74	Medium size (M)
B	32.00	297.18	Big size (M)
(Blank Row)			
#	Draft		
L	2.74		Loaded (M)
M	2.13		Mixed (M)
E	0.61		Empty (M)

Example:

SPEED
S 2.24
M 2.91
F 3.58

SIZE
L 10.67 45.72
S 10.67 178.31
M 21.34 237.74
B 32.00 297.18

DRAFT
L 2.74
M 2.13
E 0.61

P#_SAIL.DAT

Sail Line Position				
Variable	Description	Type	Format	Example
RIVER_MI	River Mile	Numeric	5.1	523.5
LSTG_LSL	Cell ID Location of Left Sailing Line at Low Stage	Character	10	305L5235
LSTG_MSL	Cell ID Location of Middle Sailing Line at Low Stage	Character	10	55L5235
LSTG_RSL	Cell ID Location of Right Sailing Line at Low Stage	Character	10	205R5235
MSTG_LSL	Cell ID Location of Left Sailing Line at Medium Stage	Character	10	305L5235
MSTG_MSL	Cell ID Location of Middle Sailing Line at Medium Stage	Character	10	55L5235
MSTG_RSL	Cell ID Location of Right Sailing Line at Medium Stage	Character	10	205R5235
HSTG_LSL	Cell ID Location of Left Sailing Line at High Stage	Character	10	305L5235
HSTG_MSL	Cell ID Location of Middle Sailing Line at High Stage	Character	10	55L5235
HSTG_RSL	Cell ID Location of Right Sailing Line at High Stage	Character	10	205R5235

Example:

715.0125L7150 85L7150 45L7150 125L7150 85L7150 45L7150 125L7150 85L7150 45L7150
716.085L7160 25L7160 65R7160 85L7160 25L7160 65R7160 85L7160 25L7160 65R7160
717.075L7170 15L7170 35R7170 75L7170 15L7170 35R7170 75L7170 15L7170 35R7170

P#_YZ.TRF

Traffic Input File				
Variable	Description	Type	Format	Example
RIVER MI	River Mile	Numeric	5.1	523.5
DIR	Upbound/Downbound	Character	1	U
SPEED	Fast/Medium/Slow	Character	1	F
SIZE	Big/Medium/Slow	Character	1	M
DRAFT	Load/Mixed/Empty	Character	1	E
TURBINE	Open Wheel/Kort Nozzle	Character	1	O
STAGE	High/Medium/Low	Character	1	L
SL POS	Left/Middle/Right	Character	1	R

Example:

715.0,U,S,S,E,K,H,M
716.0,U,S,S,E,O,H,M
717.0,U,S,S,M,K,H,M

P#_YZ.OUT

NAVEFF Output Variables				
Variable	Description	Type	Format	Example
RIVER_MI	River Mile	Numeric	5.1	523.5
DIR	Upbound/Downbound	Character	1	U
SPEED	Fast/Medium/Slow	Character	1	F
SIZE	Big/Medium/Small	Character	1	B
DRAFT	Load/Mixed/Empty	Character	1	L
TURBINE	Open Wheel/Kort Nozzle	Character	1	O
STAGE	High/Medium/Low	Character	1	H
SL_POS	Left/Middle/Right	Character	1	L
TRAFFIC	Variables 2 Thru 6 Concatenated	Character	5	UFBLO
STG_SL	Variables 7 and 8 Concatenated	Character	2	HL
CELL_ID	Cell ID	Character	10	305L5235
DEPTH	Depth (M)	Numeric	6.2	3.55
VEL_CHANGE	Maximum Velocity Change (M/S)	Numeric	7.3	0.205
DRAWDOWN	Drawdown (M)	Numeric	7.3	0.107
SL_DIST	Distance from Sailing Line (M)	Numeric	7.1	20.8
SEC_WH	Secondary Wave Height (M)	Numeric	7.3	0.107
MX_SCOUR	Maximum Scour (M)	Numeric	9.4	-0.1816
MX_SHEAR	Maximum Shear Stress (PA)	Numeric	9.4	176.2432
AMB_FLUX	Ambient Flux (Particle/Particle By Vol.)	Numeric	12.7	0.0001112
MX_AFLUX	Maximum Ambient Flux (Particle/Particle By Vol.)	Numeric	12.7	0.2848123
AMB_SHEAR	Ambient Shear Stress (PA)	Numeric	9.4	176.2432

Example:

```

328.0,D,M,S,E,O,H,L,DMSEO,HL,155L3280,3.35,0.006,0.002,-92.4,0.093,0.0000,0.8766,
0.000000,0.000000,0.8766
328.0,D,M,S,E,O,H,L,DMSEO,HL,145L3280,4.85,0.007,0.002,-82.3,0.097,-0.0001,1.2370,
0.000007,0.000007,1.2370
328.0,D,M,S,E,O,H,L,DMSEO,HL,135L3280,6.05,0.008,0.003,-71.7,0.102,-0.0001,1.5536,
0.000012,0.000012,1.5536
328.0,D,M,S,E,O,H,L,DMSEO,HL,125L3280,5.75,0.010,0.003,-61.5,0.107,-0.0001,1.4788,
0.000011,0.000011,1.4788

```

Appendix B

CHECK Program

Files

Input

P#_chan.dat

P#_elev.dat

P#_lmh.dat

P#_parm.dat

P#_sail.dat

P#_yz.trf

where: # = Pool Number

y = Stage, z = Sailing Line

Source Code

C Last change: SRJ 11 May 98 3:51 pm

C

C ==

C DESIGNED & CODED BY : Eddie Melton

C MODIFICATIONS : 12/01/97 Completed ARC/INFO File

C Transfer Code

C ==

C

C

C == Variable Declarations ==

C ==

```
CHARACTER*10 BankPos(500,4,3)
CHARACTER*15 BankPosName
CHARACTER*10 BinLabel(100000)
INTEGER BoatHalfWidth
INTEGER CenterTowLocPnt(500,3)
CHARACTER*15 CrossSectionName
INTEGER DepthError
REAL Distance(80000)
REAL Draft(4)
DOUBLE PRECISION Easting(100000)
INTEGER FatalError
REAL LastRiverMile
INTEGER LeftBankPnt(500,3)
INTEGER LeftTowLocPnt(500,3)
INTEGER MaxNumBins
INTEGER MaxNumTransects
INTEGER MaxRecords
DOUBLE PRECISION MSL(100000)
DOUBLE PRECISION Northing(100000)
INTEGER NumTransects
CHARACTER*15 ParameterName
CHARACTER*3 PoolName
INTEGER Position
INTEGER RightBankPnt(500,3)
INTEGER RightTowLocPnt(500,3)
REAL RiverMile
DOUBLE PRECISION Tabs(4,100000)
INTEGER TabsError
CHARACTER*10 TowPos(500,4,4)
CHARACTER*15 TowPosName
REAL TowSize(5,3)
INTEGER Transect(500,4)
REAL Velocity(4)
```

```

DOUBLE PRECISION WaterDepth(4,100000)
REAL      WaterLevel(500,4)
CHARACTER*1  WaterLevelId(3)
CHARACTER*15 WaterLevelName
REAL      Width(80000)

```

```

C ==
C ==      Variable Declarations      ==
C ==

```

```

C ==
C ==      Define Default Values      ==
C ==

```

```

DATA MaxNumTransects / 100 /
DATA MaxNumBins / 800 /

```

```

MaxRecords = MaxNumTransects * MaxNumBins
NumTransects = MaxNumTransects

```

```

C ==
C ==      Define Default Values      ==
C ==

```

```

C ==
C ==      Open Traffic and Output Files      ==
C ==

```

```

WRITE(*,*) 'Enter Pool Name: '
READ(*,*) PoolName

```

```

OPEN(8, FILE=PoolName//'.err', STATUS='replace')
OPEN(9, FILE=PoolName//'.out', STATUS='replace')

```

```

C ==
C ==      Open Traffic and Output Files      ==
C ==

```

```

FatalError = 0

```

```

C ==
C ==      Store Water Levels      ==
C ==

```

```

WaterLevelName = PoolName//'_lmh.dat'
OPEN(2, FILE=WaterLevelName, STATUS='old')
DO i = 1, NumTransects
  READ(2, *, END=200) RiverMile, WaterLevel(i,1),
  *      WaterLevel(i,2), WaterLevel(i,3)

```

```

        Transect(i,1) = INT(RiverMile * 10.0)
    END DO
200 CONTINUE
    WaterLevelId(1) = 'L'
    WaterLevelId(2) = 'M'
    WaterLevelId(3) = 'H'
    NumTransects = i - 1
    CLOSE(2)

C ==
C == Store Water Levels ==
C ==

C ==
C == Check Water Levels ==
C ==

DO i = 2, NumTransects
    IF (Transect(i-1,1) .GT. Transect(i,1)) THEN
        WRITE(8,*) 'River Mile Sorting Error,', WaterLevelName,',',
*           RiverMile
        FatalError = 1
    END IF
END DO

C ==
C == Check Water Levels ==
C ==

IF (FatalError .EQ. 1) GOTO 999

C ==
C == Check and Store Tow Positions ==
C ==

TowPosName = PoolName//'_sail.dat'
OPEN(3, FILE=TowPosName, STATUS='old')
DO i = 1, NumTransects
    READ(3, 350, END=300) RiverMile,
*           TowPos(i,1,1), TowPos(i,1,2), TowPos(i,1,3),
*           TowPos(i,2,1), TowPos(i,2,2), TowPos(i,2,3),
*           TowPos(i,3,1), TowPos(i,3,2), TowPos(i,3,3)
    IF (Transect(i,1) .NE. INT(RiverMile * 10.0)) THEN
        WRITE(8,*) 'River Mile Sorting Error,', TowPosName,',',
*           RiverMile
        FatalError = 1
    ENDIF
END DO
300 CONTINUE
    CLOSE(3)

```

```
350 FORMAT(f5.1, a10, a10, a10, a10, a10, a10, a10, a10)
```

```
C ==
C ==      Check and Store Tow Positions      ==
C =====
C =====
C ==
C ==      Check and Store Bank Positions      ==
C ==
```

```
BankPosName = PoolName//'_chan.dat'
OPEN(4, FILE=BankPosName, STATUS='old')
DO i = 1, NumTransects
  READ(4, 450, END=400) RiverMile,
  *      BankPos(i,1,1), BankPos(i,1,2),
  *      BankPos(i,2,1), BankPos(i,2,2),
  *      BankPos(i,3,1), BankPos(i,3,2)
  IF(Transect(i,1) .NE. INT(RiverMile * 10.0)) THEN
    WRITE(8,*) 'River Mile Sorting Error,',BankPosName,',',
  *      RiverMile
    FatalError = 1
  ENDIF
END DO
400 CONTINUE
CLOSE(4)
450 FORMAT(f5.1, a10, a10, a10, a10, a10, a10)
```

```
C ==
C ==      Check and Store Bank Positions      ==
C =====
C =====
C ==
C ==
C ==
```

```
ParameterName = PoolName//'_parm.dat'
OPEN(5, FILE=ParameterName, STATUS='old')
READ(5, *)
DO i = 1, 3
  READ(5, 550) Velocity(i)
ENDDO
READ(5, *)
READ(5, *)
DO i = 1, 4
  READ(5, 551) TowSize(i,1), TowSize(i,2)
ENDDO
READ(5, *)
READ(5, *)
DO i = 1, 3
  READ(5, 552) Draft(i)
```

```

ENDDO
CLOSE(5)

550 FORMAT(2x, f4.2)
551 FORMAT(2x, f5.2, x, f6.2)
552 FORMAT(2x, f4.2)

C ==
C ==
C ==

C ==
C ==      Read CrossSection Data      ==
C ==

CrossSectionName = PoolName/'_elev.dat'
OPEN(6, FILE=CrossSectionName, STATUS='old')
j = 0
LastRiverMile = 0.0
DO i = 1, MaxRecords
  READ(6, 650, END=600) RiverMile, BinLabel(i),
  *      Easting(i), Northing(i),
  *      MSL(i), Tabs(1,i), Tabs(2,i), Tabs(3,i)
  IF(ABS(RiverMile - LastRiverMile) .LT. 0.01) THEN
    Transect(j,3) = i
    IF(Transect(j,1) .NE. INT(RiverMile * 10.0)) THEN
      WRITE(8,*) 'River Mile Sorting Error,',CrossSectionName,',',
      *      RiverMile
      FatalError = 1
    ENDIF
  ELSE
    j = j + 1
    Transect(j,2) = i
    LastRiverMile = RiverMile
    IF(Transect(j,1) .NE. INT(RiverMile * 10.0)) THEN
      WRITE(8,*) 'River Mile Sorting Error,',CrossSectionName,',',
      *      RiverMile
      FatalError = 1
    ENDIF
  ENDIF
ENDDO
600 CONTINUE
MaxRecords = i - 1
CLOSE(6)
650 FORMAT(f5.1, a10, f12.1, f12.1, f8.1, f8.1, f8.1, f8.1)

C ==
C ==      Read CrossSection Data      ==
C ==

```


IF (FatalError .EQ. 1) GOTO 999

```
C =====
C == Check Ordering of data in the input files ==
C ==
```

```
DO i = 1, NumTransects
  IF(WaterLevel(i,1) .GT. WaterLevel(i,2)) THEN
    WRITE(8,*) 'Low Water Level > Mid Water Level,',
*      WaterLevelName,',',(Transect(i,1)/10.0)
  END IF
  IF(WaterLevel(i,2) .GT. WaterLevel(i,3)) THEN
    WRITE(8,*) 'Mid Water Level > High Water Level,',
*      WaterLevelName,',',(Transect(i,1)/10.0)
  END IF
END DO
```

```
DO i = 2, 3
  IF(Velocity(i-1) .GT. Velocity(i)) THEN
    WRITE(8,*) 'Velocity Value Out Of Order,',ParameterName
  ENDIF
  IF(Draft(i-1) .LT. Draft(i)) THEN
    WRITE(8,*) 'Draft Depths Out Of Order,',ParameterName
  ENDIF
ENDDO
```

```
DO i = 1, 4
  IF(TowSize(i,1) .GT. TowSize(i,2)) THEN
    WRITE(8,*) 'Barge Width > Barge Length,',ParameterName
  ENDIF
  IF(i .GT. 1) THEN
    IF(TowSize(i-1,1) .GT. TowSize(i,1)) THEN
      WRITE(8,*) 'Barge Width Out Of Order,',ParameterName
    ENDIF
    IF(TowSize(i-1,2) .GT. TowSize(i,2)) THEN
      WRITE(8,*) 'Barge Length Out Of Order,',ParameterName
    ENDIF
  ENDIF
ENDDO
```

```
C ==
C == Check Ordering of data in the input files ==
C =====
```

```
C =====
C == Check Existence of data in the input files ==
C ==
```

```
DO i = 1, NumTransects
  DO j = 1, 3  !Water Levels
```

```

LeftBankPnt(i,j) = 0
RightBankPnt(i,j) = 0
LeftTowLocPnt(i,j) = 0
CenterTowLocPnt(i,j) = 0
RightTowLocPnt(i,j) = 0
DO k = Transect(i,2), Transect(i,3)
  IF(BankPos(i,j,1) .EQ. BinLabel(k)) LeftBankPnt(i,j) = k
  IF(BankPos(i,j,2) .EQ. BinLabel(k)) RightBankPnt(i,j) = k
  IF(TowPos(i,j,1) .EQ. BinLabel(k)) LeftTowLocPnt(i,j) = k
  IF(TowPos(i,j,2) .EQ. BinLabel(k)) CenterTowLocPnt(i,j) = k
  IF(TowPos(i,j,3) .EQ. BinLabel(k)) RightTowLocPnt(i,j) = k
END DO
END DO
ENDDO

```

C

```

DO i = 1, NumTransects
  DO j = 1, 3 !Water Level
    IF(LeftBankPnt(i,j) .EQ. 0) THEN
      WRITE(8,*) 'Left Bank Missing',CrossSectionName,',',
*      WaterLevelId(j),',',BankPos(i,j,1)
      FatalError = 1
    ENDIF
    IF(RightBankPnt(i,j) .EQ. 0) THEN
      WRITE(8,*) 'Right Bank Missing',CrossSectionName,',',
*      WaterLevelId(j),',',BankPos(i,j,2)
      FatalError = 1
    ENDIF
    IF(LeftTowLocPnt(i,j) .EQ. 0) THEN
      WRITE(8,*) 'Left Tow Location Missing',CrossSectionName,
*      ',WaterLevelId(j),',',TowPos(i,j,1)
      FatalError = 1
    ENDIF
    IF(CenterTowLocPnt(i,j) .EQ. 0) THEN
      WRITE(8,*) 'Center Tow Location Missing',CrossSectionName,
*      ',WaterLevelId(j),',',TowPos(i,j,2)
      FatalError = 1
    ENDIF
    IF(RightTowLocPnt(i,j) .EQ. 0) THEN
      WRITE(8,*) 'Right Tow Location Missing',CrossSectionName,
*      ',WaterLevelId(j),',',TowPos(i,j,1)
      FatalError = 1
    ENDIF
  END DO
END DO

```

C

IF (FatalError .EQ. 1) GOTO 999

C

```
DO i = 1, NumTransects
  DO j = 1, 3  !Water Levels
    IF(LeftTowLocPnt(i,j) .GT. CenterTowLocPnt(i,j)) THEN
      WRITE(8,*) 'Left Tow Position > Center Tow Position,',
*         TowPosName,',',(Transect(i,1)/10.0),',',
*         WaterLevelId(j)
      FatalError = 1
    ENDIF
    IF(CenterTowLocPnt(i,j) .GT. RightTowLocPnt(i,j)) THEN
      WRITE(8,*) 'Center Tow Position > Right Tow Position,',
*         TowPosName,',',(Transect(i,1)/10.0),',',
*         WaterLevelId(j)
      FatalError = 1
    ENDIF
  END DO
END DO
```

```
DO i = 1, NumTransects
  DO j = 1, 3  !Water Levels
    IF(LeftBankPnt(i,j) .GT. RightBankPnt(i,j)) THEN
      WRITE(8,*) 'Left Bank Position > Right Bank Position,',
*         BankPosName,',',WaterLevelId(j),',',
*         BankPos(i,j,1),',',BankPos(i,j,2)
      FatalError = 1
    ENDIF
    IF(j .GT. 1) THEN
      IF(LeftBankPnt(i,j-1) .LT. LeftBankPnt(i,j)) THEN
        WRITE(8,*) 'Left Bank Position > Lower Position,',
*         BankPosName,',',WaterLevelId(j),',',
*         BankPos(i,j-1,1),',',BankPos(i,j,1)
        FatalError = 1
      END IF
      IF(RightBankPnt(i,j-1) .GT. RightBankPnt(i,j)) THEN
        WRITE(8,*) 'Right Bank Position < Lower Position,',
*         BankPosName,',',WaterLevelId(j),',',
*         BankPos(i,j-1,2),',',BankPos(i,j,2)
        FatalError = 1
      END IF
    END IF
  END DO
END DO
```

C =====

IF (FatalError .EQ. 1) GOTO 999

C =====

BoatHalfWidth = INT((TowSize(4,1) + 19.99) / 10.0) / 2

c ***** Calculate the width of each impact zone *****

DO i = 1, NumTransects

Distance(Transect(i,2)) = 0.0

Width(Transect(i,2)) = 0.0

DO j = (Transect(i,2) + 1), Transect(i,3)

Distance(j) = ABS(SQRT(

* ((Easting(Transect(i,2))-Easting(j))**2.0)+

* ((Northing(Transect(i,2))-Northing(j))**2.0)))

Width(j) = Distance(j) - Distance(j-1)

ENDDO

ENDDO

DO i = 1, NumTransects

DO j = LeftBankPnt(i,3)+1, RightBankPnt(i,3)

IF(Width(j) .GT. 17.5) THEN

WRITE(8,*) 'Impact Zone Separation > 17.5 M,'

* CrossSectionName,',',BinLabel(j)

ENDIF

IF(Width(j) .LT. 5.0) THEN

WRITE(8,*) 'Impact Zone Separation < 5.0 M,'

* CrossSectionName,',',BinLabel(j)

ENDIF

END DO

END DO

C =====

DO i = 1, NumTransects

DO j = 1, 3 !Water Levels

DO k = LeftBankPnt(i,j), RightBankPnt(i,j)

WaterDepth(j,k) = WaterLevel(i,j) - MSL(k)

END DO

END DO

END DO

DO i = 1, NumTransects

DO j = 1, 3 !Water Levels

DepthError = 0

TabError = 0

DO k = LeftBankPnt(i,j), RightBankPnt(i,j)

IF(WaterDepth(j,k) .LE. 0.0) THEN

WRITE(8,*) 'Negative Water Depth,'

```

*      CrossSectionName,',',WaterLevelId(j),',',
*      BinLabel(k),WaterDepth(j,k)
      FatalError = 1
      DepthError = 1
END IF
IF(Tabs(j,k) .LE. 0.00) THEN
  WRITE(8,*) 'Tabs Value Is Zero or Invalid,',
*      CrossSectionName,',',WaterLevelId(j),',',
*      BinLabel(k)
  TabsError = 1
END IF
END DO
IF(DepthError .EQ. 1) THEN
  k = CenterTowLocPnt(i,j)
800  CONTINUE
  IF(WaterDepth(j,k) .LE. 0.0) GOTO 810
  IF(k .EQ. RightBankPnt(i,j)) GOTO 815
  k = k + 1
  GOTO 800
810  CONTINUE
  k = k - 1
815  CONTINUE
  l = CenterTowLocPnt(i,j)
820  CONTINUE
  IF(WaterDepth(j,l) .LE. 0.0) GOTO 830
  IF(k .EQ. LeftBankPnt(i,j)) GOTO 835
  l = l - 1
  GOTO 820
830  CONTINUE
  l = l + 1
835  CONTINUE
  WRITE(9,*) 'Stage',(Transect(i,l)/10.0),',',
*      WaterLevelId(j),',',BinLabel(l),',',BinLabel(k)
END IF
IF(TabsError .EQ. 1) THEN
  k = CenterTowLocPnt(i,j)
850  CONTINUE
  IF(Tabs(j,k) .LE. 0.0) GOTO 860
  IF(k .EQ. RightBankPnt(i,j)) GOTO 865
  k = k + 1
  GOTO 850
860  CONTINUE
  k = k - 1
865  CONTINUE
  l = CenterTowLocPnt(i,j)
870  CONTINUE
  IF(Tabs(j,l) .LE. 0.0) GOTO 880
  IF(k .EQ. LeftBankPnt(i,j)) GOTO 885
  l = l - 1
  GOTO 870

```

```

880    CONTINUE
      I = I + 1
885    CONTINUE
      WRITE(9,*) 'TABS,', (Transect(i,1)/10.0),',',
*         WaterLevelId(j),',', BinLabel(l),',', BinLabel(k)
      END IF
    END DO
  END DO

```

C

```

IF (FatalError .EQ. 1) GOTO 999

```

C

```

DO i = 1, NumTransects
  DO j = 1, 3    !Water Levels
    IF(LeftTowLocPnt(i,j).LE.(LeftBankPnt(i,j)+BoatHalfWidth+1))
*   THEN
      WRITE(8,*) 'Left Tow Location To Close To Left Boundary,',
*         TowPosName,',', TowPos(i,j,1),',', WaterLevelId(j)
    END IF
    IF(RightTowLocPnt(i,j).GE.(RightBankPnt(i,j)-BoatHalfWidth-1))
*   THEN
      WRITE(8,*) 'Right Tow Location To Close To Right Boundary,',
*         TowPosName,',', TowPos(i,j,1),',', WaterLevelId(j)
    END IF
  END DO
END DO

```

C

```

DO i = 1, NumTransects
  DO j = 1, 3    !Water Levels
    DO k = 1, 3    !Sailing Lines
      IF(k .EQ. 1) Position = LeftTowLocPnt(i,j) - BoatHalfWidth
      IF(k .EQ. 2) Position = CenterTowLocPnt(i,j) - BoatHalfWidth
      IF(k .EQ. 3) Position = RightTowLocPnt(i,j) - BoatHalfWidth
      DO l = position, position+(BoatHalfWidth*2)
        IF(WaterDepth(j,l) .LT. 3.0) THEN
          WRITE(8,*) 'Tow Depth < 3 Meters,',
*             CrossSectionName,',', WaterLevelId(j),',',
*             BinLabel(l),',', WaterDepth(j,l)
          FatalError = 1
        END IF
      END DO
    END DO
  END DO
END DO

```

C =====

999 CONTINUE

CLOSE(1)

CLOSE(8)

CLOSE(9)

C =====

C == Program Termination Point ==

C == ==

STOP ' NORMAL STOP CONDITIONS '

END

C == ==

C == Program Termination Point ==

C =====

Appendix C

NAVEFF Program

Files

Input

P#_chan.dat

P#_elev.dat

P#_lmh.dat

P#_parm.dat

P#_sail.dat

P#_yz.trf

where: # = Pool Number

y = Stage, z = Sailing Line

Output

P#_yz.out

where: # = Pool Number

y = Stage, z = Sailing Line

NAVEFF Output Variables				
Variable	Description	Type	Format	Example
RIVER_MI	River Mile	Numeric	5.1	523.5
DIR	Upbound/Downbound	Character	1	U
SPEED	Fast/Medium/Slow	Character	1	F
SIZE	Big/Medium/Small	Character	1	B
DRAFT	Load/Mixed/Empty	Character	1	L
TURBINE	Open Wheel/Kort Nozzle	Character	1	O
STAGE	High/Medium/Low	Character	1	H
SL_POS	Left/Middle/Right	Character	1	L
TRAFFIC	Variables 2 Thru 6 Concatenated	Character	5	UFBLO
STG_SL	Variables 7 and 8 Concatenated	Character	2	HL
CELL_ID	Cell ID	Character	10	305L5235
DEPTH	Depth (M)	Numeric	6.2	3.55
VEL_CHANGE	Maximum Velocity Change (M/S)	Numeric	7.3	0.205
DRAWDOWN	Drawdown (M)	Numeric	7.3	0.107
SL_DIST	Distance from Sailing Line (M)	Numeric	7.1	20.8

NAVEFF Output Variables				
Variable	Description	Type	Format	Example
SEC_WH	Secondary Wave Height (M)	Numeric	7.3	0.107
MX_SCOUR	Maximum Scour (M)	Numeric	9.4	-0.1816
MX_SHEAR	Maximum Shear Stress (PA)	Numeric	9.4	176.2432
AMB_FLUX	Ambient Flux (Particle/Particle By Vol.)	Numeric	12.7	0.0001112
MX_AFLUX	Maximum Ambient Flux (Particle/Particle By Vol.)	Numeric	12.7	0.2848123
AMB_SHEAR	Ambient Shear Stress (PA)	Numeric	9.4	176.2432

Source Code

```

C   Last change: SRJ 17 Jul 98   4:25 pm
C
C   ==
C   DESIGNED & CODED BY : Eddie Melton
C   MODIFICATIONS      : 12/01/97 Completed ARC/INFO File
C                       Transfer Code
C   ==
C

```

```

C
C   ==          Variable Declarations          ==
C   ==

```

```

REAL      AmbFlux(100000)
DOUBLE PRECISION AveVel
CHARACTER*10  BankPos(500,4,3)
CHARACTER*15  BankPosName
CHARACTER*10  BinLabel(100000)
INTEGER      BinWidth
CHARACTER*15  CrossSectionName
INTEGER      CurTransect
INTEGER      Direction
CHARACTER*1   DirLoc
REAL         Distance(80000)
REAL         Draft(4)
CHARACTER*1   DraftLoc
INTEGER      DraftLocId
REAL         DrawDown(100000)
REAL         D50
REAL         D50Size(100000)
REAL         D50Vel(100000)

```

```

DOUBLE PRECISION Easting(100000)
CHARACTER*4   ErrorCode

```

INTEGER ErrorCount
 REAL LastRiverMile
 DOUBLE PRECISION LeftArea
 INTEGER LeftBankPnt
 INTEGER LeftNear
 DOUBLE PRECISION LeftWidth
 REAL LocDistance
 DOUBLE PRECISION MaxDepth
 REAL MaxFlux(100000)
 INTEGER MaxNumTransects
 INTEGER MaxNumBins
 INTEGER MaxRecords
 DOUBLE PRECISION MaxScour(100000)
 DOUBLE PRECISION NBNum
 DOUBLE PRECISION NBIncVal
 DOUBLE PRECISION MSL(100000)
 DOUBLE PRECISION Northing(100000)
 INTEGER NumTransects
 CHARACTER*15 ParameterName
 CHARACTER*1 PoolLevel
 CHARACTER*3 PoolName
 REAL VelChange(100000)
 DOUBLE PRECISION RightArea
 INTEGER RightBankPnt
 INTEGER RightNear
 DOUBLE PRECISION RightWidth
 REAL RiverMile
 REAL ShearStress(100000)
 REAL ShearAmb(100000)
 CHARACTER*1 SizeLoc
 INTEGER SizeLocID
 DOUBLE PRECISION Tabs(4,100000)
 CHARACTER*1 TowLoc
 INTEGER TowLocId
 INTEGER TowLocPnt
 CHARACTER*10 TowPos(500,4,4)
 CHARACTER*15 TowPosName
 REAL TowSize(5,3)
 CHARACTER*6 TrafficFile
 INTEGER Transect(500,4)
 REAL Velocity(4)
 CHARACTER*1 VelLoc
 INTEGER VelLocId
 DOUBLE PRECISION WaterDepth(100000)
 REAL WaterLevel(500,4)
 CHARACTER*15 WaterLevelName
 INTEGER CohSed
 INTEGER CohClass

DOUBLE PRECISION ALEFT
 DOUBLE PRECISION ALF
 DOUBLE PRECISION ALFL
 DOUBLE PRECISION ALFR
 DOUBLE PRECISION AM
 DOUBLE PRECISION ARIGHT
 DOUBLE PRECISION ATOTAL
 DOUBLE PRECISION B
 DOUBLE PRECISION BE
 DOUBLE PRECISION BLB
 DOUBLE PRECISION BLEFT
 DOUBLE PRECISION BRB
 DOUBLE PRECISION BRIGHT
 DOUBLE PRECISION BSIDE
 DOUBLE PRECISION BTOTAL
 DOUBLE PRECISION C
 DOUBLE PRECISION CF
 DOUBLE PRECISION D
 DOUBLE PRECISION DE
 DOUBLE PRECISION DEPTOW
 DOUBLE PRECISION DISP
 DOUBLE PRECISION FPV
 DOUBLE PRECISION FV
 DOUBLE PRECISION GRAV
 DOUBLE PRECISION H
 DOUBLE PRECISION L
 DOUBLE PRECISION N
 DOUBLE PRECISION NSIDEL
 DOUBLE PRECISION NSIDER
 DOUBLE PRECISION RL
 DOUBLE PRECISION RTEM
 DOUBLE PRECISION SCHJF
 DOUBLE PRECISION U1
 DOUBLE PRECISION V
 DOUBLE PRECISION VAM
 DOUBLE PRECISION VDISP
 DOUBLE PRECISION VFACTL
 DOUBLE PRECISION VFACTR
 REAL VG
 DOUBLE PRECISION VL
 DOUBLE PRECISION VLIMRAT
 DOUBLE PRECISION VLN
 DOUBLE PRECISION VLO
 REAL VNU
 DOUBLE PRECISION VW
 DOUBLE PRECISION VRAL
 DOUBLE PRECISION VRAR
 DOUBLE PRECISION VRLM
 DOUBLE PRECISION VRM
 DOUBLE PRECISION VRRM

```

real      Y
DOUBLE PRECISION yy
DOUBLE PRECISION Z
DOUBLE PRECISION ZALF
DOUBLE PRECISION ZALFL
DOUBLE PRECISION ZALFR
DOUBLE PRECISION ZC
DOUBLE PRECISION ZSL
DOUBLE PRECISION ZSM
DOUBLE PRECISION ZSML
DOUBLE PRECISION ZSMR
DOUBLE PRECISION ZSR
DOUBLE PRECISION ZT

CHARACTER*1  KOLoc
INTEGER      KOLocId

DOUBLE PRECISION A
DOUBLE PRECISION BB
DOUBLE PRECISION CDECAY
DOUBLE PRECISION CEXP
DOUBLE PRECISION CFFACTOR
DOUBLE PRECISION CFLOW
DOUBLE PRECISION CFUNC
DOUBLE PRECISION cj
DOUBLE PRECISION CJTEMP
DOUBLE PRECISION coef
DOUBLE PRECISION cp
DOUBLE PRECISION CPARA
DOUBLE PRECISION CPZ1
DOUBLE PRECISION CPZ2
DOUBLE PRECISION D0
DOUBLE PRECISION delcf
REAL          DEP
DOUBLE PRECISION DEPTMP
REAL          DP
DOUBLE PRECISION E
DOUBLE PRECISION FPX
DOUBLE PRECISION FRES
DOUBLE PRECISION FUNC
DOUBLE PRECISION FX
DOUBLE PRECISION HP
DOUBLE PRECISION HPN
DOUBLE PRECISION HPO
DOUBLE PRECISION K11
DOUBLE PRECISION KKK
DOUBLE PRECISION LBARGES
DOUBLE PRECISION nu
INTEGER       NUMX
INTEGER       NUMY

```

DOUBLE PRECISION P1DECAY
 DOUBLE PRECISION P2DECAY
 REAL PSPACE
 DOUBLE PRECISION PUSH
 DOUBLE PRECISION PUSH1
 DOUBLE PRECISION PUSH2
 DOUBLE PRECISION PUSH3
 DOUBLE PRECISION RHO
 DOUBLE PRECISION RR
 DOUBLE PRECISION s
 DOUBLE PRECISION SETBACK
 REAL TBL
 DOUBLE PRECISION temp1
 REAL THRUST
 DOUBLE PRECISION THRUSTP
 DOUBLE PRECISION TIHP
 DOUBLE PRECISION U2
 DOUBLE PRECISION V0
 DOUBLE PRECISION V1
 DOUBLE PRECISION V2
 REAL VA
 DOUBLE PRECISION VBDMAX
 DOUBLE PRECISION VDIRECT
 DOUBLE PRECISION VELLOCc
 DOUBLE PRECISION VLM(5)
 DOUBLE PRECISION VMAXTEST
 DOUBLE PRECISION VMPH
 DOUBLE PRECISION VR
 DOUBLE PRECISION VRES
 DOUBLE PRECISION VTEST
 DOUBLE PRECISION vttotal
 DOUBLE PRECISION VWAKEgx
 DOUBLE PRECISION vwakamax
 DOUBLE PRECISION VXMAX
 DOUBLE PRECISION VXRL
 DOUBLE PRECISION VXRPROP
 DOUBLE PRECISION VXRR
 DOUBLE PRECISION X
 DOUBLE PRECISION XBEGIN
 DOUBLE PRECISION XCALC
 DOUBLE PRECISION XPROP
 DOUBLE PRECISION XSPACE
 DOUBLE PRECISION YL
 DOUBLE PRECISION YR
 DOUBLE PRECISION YSPACE
 DOUBLE PRECISION ZZB

 DOUBLE PRECISION AB
 DOUBLE PRECISION alpha
 DOUBLE PRECISION SecWaveHgt(100000)

DOUBLE PRECISION HSave1
DOUBLE PRECISION HSave2

CHARACTER AKO*5

```
C ==
C ==          Variable Declarations          ==
C =====
```

```
C =====
C ==          Define Default Values          ==
C =====
```

DATA MaxNumTransects / 100 /
DATA MaxNumBins / 800 /

MaxRecords = MaxNumTransects * MaxNumBins
NumTransects = MaxNumTransects
ErrorCount = 0

```
C ==
C ==          Define Default Values          ==
C =====
```

```
C =====
C ==          Open Traffic and Output Files          ==
C =====
```

WRITE(*,*) 'Enter Traffic File Name: '
READ(*,*) TrafficFile
PoolName = TrafficFile(1:3)

OPEN(1, FILE=TrafficFile//'.trf', STATUS='old')
OPEN(8, FILE=TrafficFile//'.err', STATUS='new')
OPEN(9, FILE=TrafficFile//'.out', STATUS='new')

```
C ==
C ==          Open Traffic and Output Files          ==
C =====
```

```
C =====
C ==          Read Traffic File Header          ==
C =====
```

WRITE(8,*) 'ErrCode,Trans,Dir,Speed,Size,Draft,K/O,Stage,Sail,Traf
*fic,Stg_Sl,TotArea,LftArea,TotWth,LftWth,TowWth,TowLen,Tow Draft,V
*,Vamb,MaxDepth,HBefore,Hafter,RetVel,DrwDwn'

```

C ==
C ==      Read Traffic File Header      ==
C ==
C ==
C ==
C ==      Get Water Levels for Current River Mile      ==
C ==
C ==
      WaterLevelName = PoolName//'_lmh.dat'
      OPEN(2, FILE=WaterLevelName, STATUS='old')
      DO i = 1, NumTransects
        READ(2, *, END=200) RiverMile, WaterLevel(i,1),
        *      WaterLevel(i,2), WaterLevel(i,3)
        Transect(i,1) = INT(RiverMile * 10.0)
      END DO
200 CONTINUE
      NumTransects = i - 1
      CLOSE(2)

C ==
C ==      Get Water Levels for Current River Mile      ==
C ==
C ==
C ==
C ==      Get Tow Positions for Current River Mile      ==
C ==
C ==
      TowPosName = PoolName//'_sail.dat'
      OPEN(3, FILE=TowPosName, STATUS='old')
      DO i = 1, NumTransects
        READ(3, 350, END=300) RiverMile,
        *      TowPos(i,1,1), TowPos(i,1,2), TowPos(i,1,3),
        *      TowPos(i,2,1), TowPos(i,2,2), TowPos(i,2,3),
        *      TowPos(i,3,1), TowPos(i,3,2), TowPos(i,3,3)
      END DO
300 CONTINUE
      CLOSE(3)
350 FORMAT(f5.1, a10, a10, a10, a10, a10, a10, a10, a10, a10)

C ==
C ==      Get Tow Positions for Current River Mile      ==
C ==
C ==
C ==
C ==      Get Bank Positions for Current River Mile      ==
C ==
C ==
      BankPosName = PoolName//'_chan.dat'
      OPEN(4, FILE=BankPosName, STATUS='old')
      DO i = 1, NumTransects

```

```

      READ(4, 450, END=400) RiverMile,
*          BankPos(i,1,1), BankPos(i,1,2),
*          BankPos(i,2,1), BankPos(i,2,2),
*          BankPos(i,3,1), BankPos(i,3,2)
      END DO
400 CONTINUE
      CLOSE(4)
450 FORMAT(f5.1, a10, a10, a10, a10, a10, a10)

```

```

C ==
C ==      Get Bank Positions for Current River Mile      ==
C ==

```

```

C ==
C ==
C ==

```

```

      ParameterName = PoolName//'_parm.dat'
      OPEN(5, FILE=ParameterName, STATUS='old')
      READ(5, *)
      DO i = 1, 3
        READ(5, 550) Velocity(i)
      END DO
      READ(5, *)
      READ(5, *)
      DO i = 1, 4
        READ(5, 551) TowSize(i,1), TowSize(i,2)
      END DO
      READ(5, *)
      READ(5, *)
      DO i = 1, 3
        READ(5, 552) Draft(i)
      END DO
      CLOSE(5)
c  WRITE(*,*) 'Velocity(1)', Velocity(1)
c  WRITE(*,*) 'Velocity(2)', Velocity(2)
c  WRITE(*,*) 'Velocity(3)', Velocity(3)
c  WRITE(*,*) 'TowSize(1,x)', TowSize(1,1), TowSize(1,2)
c  WRITE(*,*) 'TowSize(2,x)', TowSize(2,1), TowSize(2,2)
c  WRITE(*,*) 'TowSize(3,x)', TowSize(3,1), TowSize(3,2)
c  WRITE(*,*) 'TowSize(4,x)', TowSize(4,1), TowSize(4,2)
c  WRITE(*,*) 'Draft(1) ', Draft(1)
c  WRITE(*,*) 'Draft(2) ', Draft(2)
c  WRITE(*,*) 'Draft(3) ', Draft(3)
550 FORMAT(2x, f4.2)
551 FORMAT(2x, f5.2, x, f6.2)
552 FORMAT(2x, f4.2)

```

```

C ==
C ==

```



```

C =====
C =====
C ==          Read CrossSection Data          ==
C ==
CrossSectionName = PoolName/'_elev.dat'
OPEN(6, FILE=CrossSectionName, STATUS='old')
j = 0
LastRiverMile = 0.0
DO i = 1, MaxRecords
c  WRITE(*,*) i
  READ(6, 650, END=600) RiverMile, BinLabel(i),
  *      Easting(i), Northing(i),
  *      MSL(i), Tabs(1,i), Tabs(2,i), Tabs(3,i),
  *      D50Size(i), D50Vel(i), CohSed, CohClass
  IF (D50Size(i) .LT. 999.0) THEN
    D50Size(i) = D50Size(i) / 1000.0
    D50Vel(i) = D50Vel(i) / 100.0
  END IF
  IF(ABS(RiverMile - LastRiverMile) .LT. 0.01) THEN
    Transect(j,3) = i
  ELSE
    j = j + 1
    Transect(j,2) = i
    LastRiverMile = RiverMile
  END IF
END DO
600 CONTINUE
MaxRecords = i - 1
CLOSE(6)
650 FORMAT(f5.1,a10,f12.1,f12.1,f8.1,f8.1,f8.1,f8.1,f8.3,f8.3,i6,i6)

C ==
C ==          Read CrossSection Data          ==
C =====
c  GOTO 149
C =====
C ==          Read Traffic File and Loop Through Algorithms          ==
C ==
DO icount = 1,99999
  ErrorCode = 'None'
  READ(1,151,END=149) RiverMile, DirLoc, VelLoc, SizeLoc,
  *      DraftLoc, KOLoc, PoolLevel, TowLoc
151  FORMAT(f5.1,x,a1,x,a1,x,a1,x,a1,x,a1,x,a1,x,a1)

C ==
C ==          Read Traffic File and Loop Through Algorithms          ==

```

```

C =====
C =====
C ==          Reformat and Change Units of Variables          ==
C ==

```

```

IF((DirLoc.EQ.'U').OR.(DirLoc.EQ.'u')) Direction = -1
IF((DirLoc.EQ.'D').OR.(DirLoc.EQ.'d')) Direction = 1
IF((VelLoc.EQ.'S').OR.(VelLoc.EQ.'s')) VelLocId = 1
IF((VelLoc.EQ.'M').OR.(VelLoc.EQ.'m')) VelLocId = 2
IF((VelLoc.EQ.'F').OR.(VelLoc.EQ.'f')) VelLocId = 3
IF((SizeLoc.EQ.'L').OR.(SizeLoc.EQ.'l')) SizeLocId = 1
IF((SizeLoc.EQ.'S').OR.(SizeLoc.EQ.'s')) SizeLocId = 2
IF((SizeLoc.EQ.'M').OR.(SizeLoc.EQ.'m')) SizeLocId = 3
IF((SizeLoc.EQ.'B').OR.(SizeLoc.EQ.'b')) SizeLocId = 4
IF((DraftLoc.EQ.'L').OR.(DraftLoc.EQ.'l')) DraftLocId = 1
IF((DraftLoc.EQ.'M').OR.(DraftLoc.EQ.'m')) DraftLocId = 2
IF((DraftLoc.EQ.'E').OR.(DraftLoc.EQ.'e')) DraftLocId = 3
IF((KOLoc.EQ.'K').OR.(KOLoc.EQ.'k')) KOLocId = 1
IF((KOLoc.EQ.'O').OR.(KOLoc.EQ.'o')) KOLocId = 2
IF((PoolLevel.EQ.'L').OR.(PoolLevel.EQ.'l')) LevelId = 1
IF((PoolLevel.EQ.'M').OR.(PoolLevel.EQ.'m')) LevelId = 2
IF((PoolLevel.EQ.'H').OR.(PoolLevel.EQ.'h')) LevelId = 3
IF((TowLoc.EQ.'L').OR.(TowLoc.EQ.'l')) TowLocId = 1
IF((TowLoc.EQ.'M').OR.(TowLoc.EQ.'m')) TowLocId = 2
IF((TowLoc.EQ.'R').OR.(TowLoc.EQ.'r')) TowLocId = 3

```

```

C ==
C ==          Reformat and Change Units of Variables          ==
C =====

```

```

DO i = 1,NumTransects
  IF(INT(RiverMile * 10.0).EQ. Transect(i,1)) GOTO 100
END DO
GOTO 859
100 CONTINUE
  CurTransect = i
  WRITE(*,*) 'Transect',RiverMile,'(',icount,')','[',ErrorCount,']

  TowLocPnt = 0
  LeftBankPnt = 0
  RightBankPnt = 0
  DO i = Transect(CurTransect,2), Transect(CurTransect,3)
    WaterDepth(i) = WaterLevel(CurTransect,LevelId) - MSL(i)
    IF(WaterDepth(i).LT. 0.0) WaterDepth(i) = 0.0
  c  *** Find Tow Location ***
    IF(TowPos(CurTransect,LevelId,TowLocId).EQ. BinLabel(i))
  *   TowLocPnt = i
    IF(BankPos(CurTransect,LevelId,1).EQ. BinLabel(i))
  *   LeftBankPnt = i

```

```

        IF(BankPos(CurTransect,LevelId,2) .EQ. BinLabel(i))
*       RightBankPnt = i
    END DO

c   DO i = LeftBankPnt, RightBankPnt
c       IF(WaterDepth(i) .EQ. 0.0) WRITE(8,*) BinLabel(i),',',LevelId
c   END DO

    IF(TowLocPnt .EQ. 0) GOTO 860
    IF(LeftBankPnt .EQ. 0) GOTO 861
    IF(RightBankPnt .EQ. 0) GOTO 862

c   *** Calculate Areas ***
    LeftArea = 0.0
    RightArea = 0.0
    DO i = TowLocPnt, LeftBankPnt, -1
        LeftArea = LeftArea + (WaterDepth(i) * 10.0)
    END DO
    DO i = TowLocPnt, RightBankPnt
        RightArea = RightArea + (WaterDepth(i) * 10.0)
    END DO
c   *** Adjust Areas for the Tow Location ***
    LeftArea = LeftArea - (WaterDepth(TowLocPnt) * 10.0 / 2.0)
    RightArea = RightArea - (WaterDepth(TowLocPnt) * 10.0 / 2.0)

    AveVel = 0.0
    MaxDepth = 0.0
    DO i = LeftBankPnt, RightBankPnt
        AveVel = AveVel + (Tabs(LevelId,i) * (WaterDepth(i) * 10.0))
        IF(WaterDepth(i) .GT. MaxDepth) MaxDepth = WaterDepth(i)
        Distance(i) = ABS(SQRT(((Easting(TowLocPnt)-Easting(i))**2.0)
*           + ((Northing(TowLocPnt)-Northing(i))**2.0)))
    END DO
    AveVel = AveVel / (LeftArea + RightArea)

    LeftWidth = SQRT(
*       ((Easting(TowLocPnt)-Easting(LeftBankPnt)) ** 2.0) +
*       ((Northing(TowLocPnt)-Northing(LeftBankPnt)) ** 2.0))
    RightWidth = SQRT(
*       ((Easting(TowLocPnt)-Easting(RightBankPnt)) ** 2.0) +
*       ((Northing(TowLocPnt)-Northing(RightBankPnt)) ** 2.0))

    BinWidth = INT(TowSize(SizeLocId,1) / 10.0)
    LeftNear = TowLocPnt - BinWidth
    RightNear = TowLocPnt + BinWidth

c   GOTO 99
C   =====
C   ==      Start of SCHIJF Method from Visual Basic Code      ==

```

C ==

==

BTOTAL = LeftWidth + RightWidth
BLEFT = LeftWidth
ATOTAL = LeftArea + RightArea
ALEFT = LeftArea

D = Draft(DraftLocId)
B = TowSize(SizeLocId,1)
L = TowSize(SizeLocId,2)
DEPTOW = WaterDepth(TowLocPnt)

IF(DEPTOW .LT. 3.0) GOTO 854

VAM = AveVel
VW = Velocity(VelLocId)
VG = VW + VAM * Direction
V = ABS(VG - (Direction * 1.2 * VAM))
GRAV = 9.805

c *** SET WATER VISCOSITY = 0.0000011 M**2/SEC FOR TEMP = 17 DEG C
VNU = 0.0000011

c *** COMPUTE GEOMETRIC FACTORS

AM = B * D
BRIGHT = BTOTAL - BLEFT
ARIGHT = ATOTAL - ALEFT
NSIDEL = 2.0 * ALEFT / AM
NSIDER = 2.0 * ARIGHT / AM
BLB = BLEFT / BTOTAL
BRB = BRIGHT / BTOTAL
H = ATOTAL / BTOTAL
HSave1 = H
N = ATOTAL / AM
IF((H / MaxDepth) .GT. 0.666) GOTO 21
H = H * (3.0 - (3.0 * H / MaxDepth))

21 CONTINUE

HSave2 = H

c SOLVE HOCHSTEIN EQUATION FOR U1 FOR DISPLACEMENT CALCULATION
U1 = ABS(V) * ((N / (N-1)) ** 1.25-1)

c *** COMPUTE DISPLACEMENT THICKNESS

50 continue

VDISP = ABS(V) + U1
RL = VDISP * L / VNU
DISP = 0.292 * L / (0.43429 * LOG(ABS(RL))) ** 2.58
DE = D + DISP
BE = B + 2.0 * DISP
N = ATOTAL / BE / DE

c *** SOLVE SCHIJF EQUATION FOR LIMIT SPEED USING NEWTON RAPHSON

LoopCnt = 1

VLO = ABS(V)

54 continue

RTEM = VLO ** 2.0 / GRAV / H

FV = 1.0 - 1.0 / N + 0.5 * RTEM - 1.5 * RTEM ** (1.0 / 3.0)

FPV = VLO / GRAV / H - (VLO ** (-1.0 / 3.0)) /

* (GRAV * H) ** (1.0 / 3.0)

VLN = VLO - FV / FPV

IF (ABS((VLO - VLN) / VLO) .LT. 0.0001) GOTO 55

LoopCnt = LoopCnt + 1

VLO = VLN

IF (LoopCnt .GE. 100) GOTO 853

GOTO 54

55 continue

VL = VLN

C THIS ROUTINE FINDS U1 FOR V > .95*VL

IF (V .LT. 0.95*VL) GOTO 57

VTEMP = .95 * VL

Z = .01

56 SCHIJF = (1 + N * Z / H) / (N - 1 - N * Z / H)

ZT = (VTEMP ** 2 / 2 / GRAV) * ((SCHIJF ** 2) + 2 * SCHIJF)

U1 = ABS(VTEMP) * SCHIJF

IF (ABS((ZT - Z) / ZT) .LT. 0.00001) GOTO 45

Z = ZT

GOTO 56

45 VRRAT = U1 / VTEMP

U1 = V * VRRAT

GOTO 52

c *** SOLVE SCHIJF EQUATION FOR RETURN VELOCITY

57 LoopCnt = 1

Z = 0.01

51 SCHIJF = (1.0 + N * Z / H) / (N - 1.0 - N * Z / H)

ZT = (V ** 2.0 / 2.0 / GRAV) * ((SCHIJF ** 2.0) + 2.0 * SCHIJF)

U1 = ABS(V) * SCHIJF

IF (ABS((ZT - Z) / ZT) .LT. 0.00001) GOTO 52

LoopCnt = LoopCnt + 1

Z = ZT

IF (LoopCnt .GE. 100) GOTO 852

GOTO 51

52 CONTINUE

VLIMRAT = ABS(V) / VL

c *** APPLY CORRECTION FACTOR

61 continue

CF = 1.78 - 1.07 * VLIMRAT

IF (CF .LT. 1.0) CF = 1.0

U1 = CF * U1

ZT = (ABS(V) + U1) ** 2.0 / 2.0 / GRAV - V ** 2.0 / 2.0 / GRAV

c *** COMPUTE a(ALF) AND AVERAGE Vr FOR EACH SIDE OF TOW

VFACTL = 1.65 - 1.3 * BLB

IF (BLB .GT. 0.5) VFACTL = 1.35 - 0.7 * BLB

VFACTR = 1.65 - 1.3 * BRB

IF (BRB .GT. 0.5) VFACTR = 1.35 - 0.7 * BRB

VRAL = U1 * VFACTL

VRAR = U1 * VFACTR

ZSL = ZT * VFACTL

ZSR = ZT * VFACTR

ALFL = 0.75 * NSIDEL ** 0.18

ALFR = 0.75 * NSIDER ** 0.18

IF (ALFL .LT. 1.0) ALFL = 1.0

IF (ALFR .LT. 1.0) ALFR = 1.0

ZALFL = ALFL ** 0.5

ZALFR = ALFR ** 0.5

VRLM = ALFL * VRAL

VRRM = ALFR * VRAR

ZSML = ZALFL * ZSL

ZSMR = ZALFR * ZSR

IF ((ARIGHT .GT. ALEFT) .AND. (VRRM .GT. VRLM)) VRRM = VRLM

IF ((ALEFT .GT. ARIGHT) .AND. (VRLM .GT. VRRM)) VRLM = VRRM

c *** Print input parameters

c WRITE(9,881) ATOTAL, ALEFT

c WRITE(9,882) BTOTAL, BLEFT

c WRITE(9,883) B, D

c WRITE(9,884) L

c WRITE(9,885) VG

c WRITE(9,886) VAM

c *** COMPUTE RETURN VELOCITY AND DRAWDOWN DISTRIBUTION

ALF = ALFL

VRM = VRLM

ZALF = ZALFL

ZSM = ZSML

BSIDE = BLEFT

C = 3.0 * LOG(1.0 / ALF)

ZC = 3.0 * LOG(1.0 / ZALF)

DO i = LeftBankPnt, LeftNear

y = Distance(i) * (-1.0)

yy = ABS(Y)

VelChange(i) = VRM * EXP(C * (yy - B) / (BSIDE - B))

DrawDown(i) = ZSM * EXP(ZC * (yy - B) / (BSIDE - B))

END DO

```

ALF = ALFR
VRM = VRRM
ZALF = ZALFR
ZSM = ZSMR
BSIDE = BRIGHT
C = 3.0 * LOG(1.0 / ALF)
ZC = 3.0 * LOG(1.0 / ZALF)
DO i = RightNear, RightBankPnt
  y = Distance(i)
  yy = ABS(Y)
  VelChange(i) = VRM * EXP(C * (yy - B) / (BSIDE - B))
  DrawDown(i) = ZSM * EXP(ZC * (yy - B) / (BSIDE - B))
END DO

```

```

C ==
C ==      Start of SCHIJF Method from Visual Basic Code      ==
C ==

```

```

C ==
C ==      Secondary Wave Height      ==
C ==

```

```

DO i = LeftBankPnt, RightBankPnt
  SecWaveHgt(i) = 0.0
  LocDistance = TowSize(SizeLocId,1) / 2.0 + 14.0
  IF(Distance(i) .GT. LocDistance) THEN
    AB = TowSize(SizeLocId,1) * Draft(DraftLocId)
    IF(AB .LE. 30.0) THEN
      alpha = 0.5
    ELSE
      IF(AB .LE. 65.0) THEN
        alpha = 0.6
      ELSE
        alpha = 0.7
      END IF
    END IF
    SecWaveHgt(i) = alpha * ((Distance(i) - (TowSize(SizeLocId,1)
*      / 2.0)) ** (-1.0/3.0)) * ((Velocity(VelLocId)
*      / SQRT(GRAV)) ** 2.67)
    IF((SecWaveHgt(i) / WaterDepth(i)) .GT. 0.6) THEN
      SecWaveHgt(i) = SecWaveHgt(i) * (-1.0)
    END IF
  ELSE
    SecWaveHgt(i) = -9.0
  END IF
END DO

```

```

C ==
C ==      Secondary Wave Height      ==
C ==

```

```

C =====
C == Start of Propeller Jet Velocities from Visual Basic Code ==
C ==

```

```

VAM = AveVel
VW = Velocity(VelLocId)
VG = VW * Direction + VAM
B = TowSize(SizeLocId,1)
D = Draft(DraftLocId)
LBARGES = TowSize(SizeLocId,2)
VDIRECT = Direction

```

c THE FOLLOWING 2 VALUES ARE TYPICAL OF UMRS TOWS AND WOULD BE VERY

c DIFFICULT TO DETERMINE THEIR ACTUAL VALUES FOR ALL TOWBOATS
 TBL = 52.0 ! TOWBOAT LENGTH
 SETBACK = 5.0 ! DISTANCE FROM PROP TO TOWBOAT STERN(16.4 FT OR 5 M)
 VA = VAM ! enter ambient AVG CHANNEL velocity
 DEP = WaterDepth(TowLocPnt)
 VR = U1 ! AVERAGE RETURN VELOCITY FROM NAVEFF
 z = ZT ! AVERAGE DRAWDOWN FROM NAVEFF

c *** VEL PREDICTION AT 0.46 m ABOVE CHANNEL BOTTOM TO AGREE WITH MEASUREMENTS

VELLOC = 0.46

c BC = DEP - D ! BOTTOM CLEARANCE USED IN WAKE FLOW COMPUTATIONS

grav = 9.805

RHO = 999.8

c *** Compute thrust using PROGRAM "power.BAS" relations

c *** program assumes a semi-integrated tow (FRES = 1.0)

FRES = 1.0

nu = 0.00000112 ! ABOUT 16 DEG C

delcf = 0.00075

cp = 0.20

c *** computations

vtotal = VW + VR

s = 2.0 * D * LBARGES + LBARGES * B ! HULL AREA EXPOSED TO WATER

c *** COMPUTE SKIN FRICTION COEFFICIENT FOR DEEP WATER

CF = (0.075 * (((LOG10(ABS(vtotal * LBARGES / nu)))) -

* 2.0) ** (-2.0))) + delcf

c *** INCREASE HULL SHEAR FOR LOW UKC

CFFACTOR = 1.55 - 0.105 * DEP / D

IF(CFFACTOR .LT. 1.0) CFFACTOR = 1.0

CFLOW = CF * CFFACTOR

c *** COMPUTE RESISTANCE USING VAN DE KAA (1978)


```

PUSH1 = CFLOW * 0.5 * RHO * vtotal ** 2.0 * s
PUSH2 = RHO * grav * B * D * z
PUSH3 = cp * 0.5 * RHO * VW ** 2.0 * B * D
PUSH = PUSH1 + PUSH2 + PUSH3

c  *** ADJUST RESISTANCE FOR TOW INTEGRATION (ASSUMED TO BE SEMI-
INTEGRATED)
    THRUST = FRES * PUSH
c  PRINT "TOTAL THRUST (NEWTONS) = ", THRUST

c  *** TOUTANT POWER USING NEWTON-RAPHSON
IF(KOLocId .EQ. 1) THEN
    A = 31.82
    BB = 5.4
ELSE
    A = 23.57
    BB = 2.3
END IF

VMPH = ABS(VA - VG) * 3.28 * 60.0 / 88.0
THRUSTP = THRUST / 4.4482
V1 = THRUSTP / A
V2 = (BB / A) * VMPH ** 2.0
HPO = 500.0
KKK = 0.0
15  FX = HPO ** 0.974 - V1 - V2 * HPO ** 0.5
    FPX = 0.974 * HPO ** (-0.026) - 0.5 * V2 * HPO ** (-0.5)
    HPN = HPO - FX / FPX
    IF(ABS(HPO - HPN) .LT. 0.1) GOTO 19
    IF(KKK .GT. 50.0) GOTO 80
    HPO = HPN
    KKK = KKK + 1.0
    GOTO 15
19  CONTINUE

c  *** SET TOTAL INSTALLED HORSEPOWER = 1.2*APPLIED HP
    TIHP = 1.2 * HPN

c  *** COMPUTE PROPELLER DIAMETER USING REGRESSION EQUATIONS
IF(KOLocId .EQ. 1) DP = ((6.3 * TIHP ** 0.33) / 12.0 / 3.28)
IF(KOLocId .EQ. 2) DP = ((5.25 * TIHP ** 0.35) / 12.0 / 3.28)
IF(DP .GT. 2.80) DP = 2.80 ! LIMITS DP TO 9.5 FT
IF(DP .LT. 1.80) DP = 1.80 ! LIMITS DP TO 6 FT
c  PRINT "PROPELLER DIAMETER (METERS) = ", DP
    PSPACE = 2.19 * DP ! SETS DIST BETWEEN PROPS BASED ON DP

c  *** SET HP = DEPTH MINUS 1/2 PROP DIAMETER
    HP = DEP - DP / 2.0
    THRUST = THRUST / 2.0 ! CONVERTS TO THRUST PER PROPELLER

```

```

c  *** X IS MEASURED FROM BOW OF BARGES
    XBEGIN = LBARGES + TBL - 20.0 !BEGIN LOOKING FOR MAX 20M AHEAD OF
    STERN
    XSPACE = 1.9
    NUMX = 200

c  *** Y IS MEASURED Laterally FROM CENTER OF TOWBOAT
    Y = 3.0 !Y=0 M IS REPRESENTED BY VEL AT Y=3 M BECAUSE PEAK AT 3 M
c  *** CAN BE SUBSTANTIALLY GREATER THAN AT Y=0 AT SHALLOW DEPTHS
    YSPACE = 10.0
    NUMY = 3

c  *** MISCELLANEOUS INPUT

c  ***INPUT APPLICABLE TO BOTH PROPELLER TYPES
    CDECAY = 0.34
    P1DECAY = 0.93
    P2DECAY = 0.24

    IF(KOLocId .EQ. 1) GOTO 70
c  *** THIS SECTION FOR SETTING OPEN WHEEL PARAMETERS
    D0 = 0.71 * DP
    E = 0.43
    CPARA = 0.12 * (DP / HP) ** 0.6666
    CEXP = 0.656
    CFUNC = 0.5
    GOTO 190
70 CONTINUE
c  *** THIS SECTION FOR SETTING KORT NOZZLE PARAMETERS
    D0 = DP
    E = 0.58
    CPARA = 0.04
    CEXP = 0.85
    CFUNC = 0.25
c  *** END OF KORT VERSUS OPEN
190 CONTINUE

c  *** end input

c  *** COMPUTE VELOCITY EXITING PROPELLER
    V0 = 1.13 / D0 * (THRUST / RHO) ** 0.5
    U2 = V0
    IF(U2 .EQ. 0.0) U2 = 0.00001
c  PRINT "VELOCITY EXITING PROPELLER (METERS/SEC) = ", U2

c  *** BEGIN ITERATION LOOP FOR X AND Y
    DO J = 1, NUMY
        IF(J .EQ. 2) Y = 10.0 !THIS RESETS LOCATION TO 10 M INCREMENTS
        X = XBEGIN !x = 0 at bow of barges
        VTEST = 0.0

```

```

DO i = 1, NUMX

c   *** COMPUTE PEAK VELOCITY CHANGE AT BOW
VBDMAX = (VA - VG) * 0.79 * (DEP / D) ** (-1.21)
IF(Y .GT. (B / 2.0)) VBDMAX = 0.0
IF(ABS(VBDMAX) .GT. VTEST) VTEST = ABS(VBDMAX)

c   *** compute wake velocity
vwakamax = (-1.0) * (VA - VG) * 0.78 * (D / DEP) ** 1.81
VWAKEgx = 0.0
IF((X - LBARGES) .GT. TBL) GOTO 30
coef = X - LBARGES
IF(coef .LT. 0.0) coef = 0.0
IF(Y .GT. (B / 2.0)) coef = 0.0
VWAKEgx = vwakamax * coef / TBL
GOTO 39

30  CONTINUE
temp1 = (1 + 0.0075 * (TBL / D) - 0.0075 * (X - LBARGES) / D)
IF(temp1 .LT. 0.0) temp1 = 0.0
IF(Y .GT. (B / 2.0)) temp1 = 0.0
VWAKEgx = vwakamax * temp1

c   *** END WAKE VEL
39  CONTINUE

c   *** BEGIN PROPELLOR JET VELOCITY
XPROP = X - LBARGES - TBL + SETBACK !x relative to props
VXRPROP = 0.0
IF(XPROP .LT. 0.0) GOTO 69 !GOES TO END OF PROPELLER

c   *** COMPUTE VELOCITY IN ZONE 1 WHICH IS TWO JETS ADDED TOGETHER

c   *** DECAY MAX JET VELOCITY USING SINGLE JET EQUATION
XCALC = XPROP
IF(XPROP .LT. (2.03 * DP)) XCALC = 2.03 * DP
VXMAX = U2 * 1.45 * (XCALC / DP) ** (-0.524)
IF(VXMAX .GT. U2) VXMAX = U2

c   *** COMPUTE LOCATION OF PARABOLIC JET OFF RUDDER
c   *** CJ IS THE LOCATION OF THE CENTER OF THE JET RELATIVE TO SHAFT
CJTEMP = Cpara * grav * (XPROP-SETBACK/2.0) ** 2.0 / U2 ** 2.0
*      / 0.957
cj = (-1.0) * (0.2126 * (XPROP - SETBACK / 2.0) - CJTEMP)
ZZB = HP + cj - VELLOCc !ZZB IS CENTER OF JET RELATIVE TO VELLOC
IF(XPROP .LT. (SETBACK / 2.0)) ZZB = HP - VELLOCc !THIS IS BEFORE JET
DEFLECTED
IF(XPROP .LT. (SETBACK / 2.0)) GOTO 601 !OFF OF THE RUDDER
IF(ZZB .GT. (DEP - VELLOCc)) GOTO 500 !THIS DEFINES END OF ZONE 1
WHERE
c   PARABOLIC SHAPE GOES TO SURFACE
601  CONTINUE

```

```

YL = Y + PSPACE / 2.0
YR = Y - PSPACE / 2.0
RL = SQRT(YL ** 2.0 + (ZZB) ** 2.0)
RR = SQRT(YR ** 2.0 + (ZZB) ** 2.0)
CPZ1 = 0.18
VXRL = VXMAX * EXP((-1.0) * (RL) ** 2.0 / (2.0 *
*   (CPZ1) ** 2.0 * (XPROP) ** 2.0))
VXRR = VXMAX * EXP((-1.0) * (RR) ** 2.0 / (2.0 *
*   (CPZ1) ** 2.0 * (XPROP) ** 2.0))
VXRPROP = VXRR + VXRL

c   *** THIS LIMITS PROP VEL IN ZONE 1 TO FUEHRER, ROMISCH, ENGELKE
TYPE EQ

VMAXTEST = E * (DP / HP) * U2
IF(VXRPROP .GT. VMAXTEST) VXRPROP = VMAXTEST

GOTO 69 !THIS SKIPS ZONE 2 CALC BECAUSE STILL IN ZONE 1

c   *** COMPUTE MAX PROP VEL IN ZONE 2
500  CONTINUE
VXMAX = U2 * CEXP * 2.7183 ** (-0.0178 * XPROP / DP)

c   *** COMPUTE LATERAL DISTRIBUTION OF MAXIMUM WHICH IS AT
SURFACE

CPZ2 = 0.84 * (XPROP / DP) ** (-0.62)
VXRPROP = VXMAX * EXP(-Y ** 2.0 / (2.0 * (CPZ2) ** 2.0 *
*   (XPROP) ** 2.0))

c   *** COMPUTE DECAY FROM SURFACE TO BOTTOM
K11 = CDECAY * (DP / HP) ** P1DECAY * (XPROP / DP) ** P2DECAY
IF(K11 .GT. 0.95) K11 = 0.95
VXRPROP = VXRPROP * K11

c   *** SUM OF VPROP, VWAKE
69  CONTINUE
FUNC = 1.0 - CFUNC * ABS((VA - VG) / U2) * (HP / DP) ** 1.5
IF(FUNC .LT. 0.0) FUNC = 0.0
VRES = (-1.0) * VDIRECT * VXRPROP * FUNC + VWAKEgx
IF(ABS(VRES) .GT. VTEST) VTEST = ABS(VRES)

X = X + XSPACE
END DO

VLM(J) = VTEST
Y = Y + YSPACE
END DO

```

```

c *** END ITERATION LOOP ON X AND Y

c *** OUTPUT

c PRINT "CL DIST, M MAX VEL CHANGE, M/SEC"
c FOR K = 1 TO NUMY
c YY(1) = 0
c PRINT YY(K), VLM(K)
c NEXT K
c PRINT "TOTAL INSTALLED HORSEPOWER (=1.2*APPLIED POWER) = ", TIHP

```

```

80 CONTINUE

```

```

C ==
C == Start of Propeller Jet Velocities from Visual Basic Code ==
C ==

```

```

DO i = LeftBankPnt, (TowLocPnt-1)
  Distance(i) = Distance(i) * (-1.0)
END DO

```

```

NBNum = RightNear - LeftNear + 2.0
NBIncVal = (DrawDown(RightNear+1) - DrawDown(LeftNear-1)) / NBNum
DO i = LeftNear, RightNear
  DrawDown(i) = DrawDown(i-1) + NBIncVal
END DO

```

```

NBNum = RightNear - LeftNear + 2.0
NBIncVal = (VelChange(RightNear+1) - VelChange(LeftNear-1)) / NBNum
DO i = LeftNear, RightNear
  VelChange(i) = VelChange(i-1) + NBIncVal
END DO

```

```

IF(VLM(1) .GT. VelChange(TowLocPnt))
* VelChange(TowLocPnt) = VLM(1)
IF(VLM(2) .GT. VelChange(TowLocPnt-1))
* VelChange(TowLocPnt-1) = VLM(2)
IF(VLM(2) .GT. VelChange(TowLocPnt+1))
* VelChange(TowLocPnt+1) = VLM(2)

```

```

c IF(VelChange(TowLocPnt-1) .LT. VelChange(LeftNear-1))
c * VelChange(TowLocPnt-1) = VelChange(LeftNear-1)
c IF(VelChange(TowLocPnt+1) .LT. VelChange(RightNear+1))
c * VelChange(TowLocPnt+1) = VelChange(RightNear+1)
c IF(VelChange(TowLocPnt) .LT. VelChange(TowLocPnt-1))
c * VelChange(TowLocPnt) = VelChange(TowLocPnt-1)
c IF(VelChange(TowLocPnt) .LT. VelChange(TowLocPnt+1))
c * VelChange(TowLocPnt) = VelChange(TowLocPnt+1)

```

```

c***** Interpolation from Tow RVel to Original NAVEFF RVels *****

```

```

c IF(VLM(2) .EQ. 0.0) THEN
c   NBNuM = (TowLocPnt-1) - LeftNear + 2.0
c   NBIncVal = (VelChange(TowLocPnt)-VelChange(LeftNear-1))/NBNuM
c   DO i = LeftNear, (TowLocPnt-1)
c     VelChange(i) = VelChange(i-1) + NBIncVal
c   END DO
c
c   NBNuM = RightNear - (TowLocPnt+1) + 2.0
c   NBIncVal = (VelChange(RightNear+1)-VelChange(TowLocPnt))/NBNuM
c   DO i = (TowLocPnt+1), RightNear
c     VelChange(i) = VelChange(i-1) + NBIncVal
c   END DO
c ELSE
c   NBNuM = (TowLocPnt-2) - LeftNear + 2.0
c   NBIncVal = (VelChange(TowLocPnt-1)-VelChange(LeftNear-1))/NBNuM
c   DO i = LeftNear, (TowLocPnt-2)
c     VelChange(i) = VelChange(i-1) + NBIncVal
c   END DO
c
c   NBNuM = RightNear - (TowLocPnt+2) + 2.0
c   NBIncVal = (VelChange(RightNear+1)-VelChange(TowLocPnt+1))/NBNuM
c   DO i = (TowLocPnt+2), RightNear
c     VelChange(i) = VelChange(i-1) + NBIncVal
c   END DO
c ENDIF

C =====
C ==      Initialize Variables and Call Scour Routines      ==
C ==                                                    ==

IF(KOLocId .EQ. 1) AKO = "k"  ! kort or open
IF(KOLocId .EQ. 2) AKO = "o"  ! kort or open
C   DP = 2.74    ! propeller diameter
vg = (Velocity(VelLocId) * Direction) + AveVel ! vessel speed relative to ground
DIRECT = Direction    ! -1 = upbound, +1 = downbound
barbeam = TowSize(SizeLocId,1) ! total width of barges
DRAFTc = Draft(DraftLocId)    ! draft of barges
barlen = TowSize(SizeLocId,2) ! total length of barges
tbl = 52.0    ! length of towboat- keep constant for all tows
C   THRUST = 300000.0 ! thrust for each props in newtons
VNU = .000001 ! kinematic viscosity- fix for all conditions????
Ylamb = .3    ! porosity of sediment- fix for all ??????????????

c*****
c////////////////////////////////////////\////////////////////////////////
c////////////////////////////////////////\////////////////////////////////
c*****

```

iTRANTYPE = 2 ! ACKER-WHITE(1) OR GARCIA(2) TRANSPORT

```
va = AveVel          ! ambient average channel velocity
dep = WaterDepth(TowLocPnt) ! local depth at centerline of tow
deptmp = WaterDepth(TowLocPnt)
if ( deptmp .LT. 3.5 ) dep = 3.5

D50 = D50Size(TowLocPnt) ! SEDIMENT DIAMETER IN M
VS = D50Vel(TowLocPnt) ! SEDIMENT FALL VELOCITY IN M/SEC
Y = 3.0 ! distance from centerline for near vessel scour only
VRMAX = VelChange(TowLocPnt) ! max return vel at point of interest
VACell = Tabs(LevelId,TowLocPnt)
VABOTT = 0.7 * VACell ! bottom velocity- set equal to .7*va
deltime = .1 ! time step for computations- fix at .1 unless too slow
IF( D50 .LT. 999.0 ) THEN
  call propscou(ako, dp, vg, va, direct, barbeam, draftc, barlen,
*          tbl, thrust, dep, vabott, d50, y, ETAMAX, VNU,
*          YLAMB, VS, deltime, iTRANTYPE, TauMax, PSPACE,
*          VACell,CNewMax,Camb, TauAmb)
  MaxScour(TowLocPnt) = ETAMAX
  ShearStress(TowLocPnt) = TauMax
  AmbFlux(TowLocPnt) = CAMB
  MaxFlux(TowLocPnt) = CNEWMAX
  ShearAmb(i) = TauAmb
ELSE
  MaxScour(TowLocPnt) = 9999.0
  ShearStress(TowLocPnt) = 9999.0
  AmbFlux(TowLocPnt) = 9999.0
  MaxFlux(TowLocPnt) = 9999.0
  ShearAmb(i) = 9999.0
ENDIF
```

```
va = AveVel          ! ambient average channel velocity
dep = WaterDepth(TowLocPnt) ! local depth at centerline of tow
deptmp = WaterDepth(TowLocPnt)
if ( deptmp .LT. 3.5 ) dep = 3.5
```

```
D50 = D50Size(TowLocPnt-1) ! SEDIMENT DIAMETER IN M
VS = D50Vel(TowLocPnt-1) ! SEDIMENT FALL VELOCITY IN M/SEC
Y = ABS(Distance(TowLocPnt-1)) ! distance from centerline for near vessel scour only
VRMAX = VelChange(TowLocPnt-1) ! max return vel at point of interest
VACell = Tabs(LevelId,TowLocPnt-1)
VABOTT = 0.7 * VACell ! bottom velocity- set equal to .7*va
deltime = .1 ! time step for computations- fix at .1 unless too slow
IF( D50 .LT. 999.0 ) THEN
  call propscou(ako, dp, vg, va, direct, barbeam, draftc, barlen,
*          tbl, thrust, dep, vabott, d50, y, ETAMAX, VNU,
*          YLAMB, VS, deltime, iTRANTYPE, TauMax, PSPACE,
*          VACell,CNewMax,Camb, TauAmb)
  MaxScour(TowLocPnt-1) = ETAMAX
```

```

ShearStress(TowLocPnt-1) = TauMax
AmbFlux(TowLocPnt-1) = CAMB
MaxFlux(TowLocPnt-1) = CNEWMAX
ShearAmb(i) = TauAmb
ELSE
  MaxScour(TowLocPnt-1) = 9999.0
  ShearStress(TowLocPnt-1) = 9999.0
  AmbFlux(TowLocPnt-1) = 9999.0
  MaxFlux(TowLocPnt-1) = 9999.0
  ShearAmb(i) = 9999.0
ENDIF

va = AveVel          ! ambient average channel velocity
dep = WaterDepth(TowLocPnt) ! local depth at centerline of tow
deptmp = WaterDepth(TowLocPnt)
if ( deptmp .LT. 3.5 ) dep = 3.5

D50 = D50Size(TowLocPnt+1) ! SEDIMENT DIAMETER IN M
VS = D50Vel(TowLocPnt+1) ! SEDIMENT FALL VELOCITY IN M/SEC
Y = ABS(Distance(TowLocPnt+1)) ! distance from centerline for near vessel scour only
VRMAX = VelChange(TowLocPnt+1) ! max return vel at point of interest
VACell = Tabs(LevelId,TowLocPnt+1)
VABOTT = 0.7 * VACell ! bottom velocity- set equal to .7*va
deltime = .1 ! time step for computations- fix at .1 unless too slow
IF( D50 .LT. 999.0 ) THEN
  call propscou(ako, dp, vg, va, direct, barbeam, draftc, barlen,
*          tbl, thrust, dep, vabott, d50, y, ETAMAX, VNU,
*          YLAMB, VS, deltime, iTRANSTYPE, TauMax, PSPACE,
*          VACell,CNewMax,Camb,TauAmb)
  MaxScour(TowLocPnt+1) = ETAMAX
  ShearStress(TowLocPnt+1) = TauMax
  AmbFlux(TowLocPnt+1) = CAMB
  MaxFlux(TowLocPnt+1) = CNEWMAX
  ShearAmb(i) = TauAmb
ELSE
  MaxScour(TowLocPnt+1) = 9999.0
  ShearStress(TowLocPnt+1) = 9999.0
  AmbFlux(TowLocPnt+1) = 9999.0
  MaxFlux(TowLocPnt+1) = 9999.0
  ShearAmb(i) = 9999.0
ENDIF

DO i = LeftBankPnt, TowLocPnt-2
  va = Tabs(LevelId,i) ! ambient average channel velocity
  dep = WaterDepth(i) ! local depth at centerline of tow
  D50 = D50Size(i) ! SEDIMENT DIAMETER IN M
  VS = D50Vel(i) ! SEDIMENT FALL VELOCITY IN M/SEC
  Y = ABS(Distance(i)) ! distance from centerline for near vessel scour only
  VRMAX = VelChange(i) * direct * (-1.0) ! max return vel at point of interest
  VACell = Tabs(LevelId,i)

```



```

VABOTT = 0.7 * VACell ! bottom velocity- set equal to .7*va
deltime = .1 ! time step for computations- fix at .1 unless too slow
IF( D50 .LT. 999.0 ) THEN
  CALL VRSCOUR(VG,VRMAX,VA,DEP,BARLEN,DELTIME,D50,VS,VNU,YLAMB,
*      ETAMAX,iTRANTYPE,TauMax,VACell,CNewMax,Camb,
*      TauAmb)
  MaxScour(i) = ETAMAX
  ShearStress(i) = TauMax
  AmbFlux(i) = CAMB
  MaxFlux(i) = CNEWMAX
  ShearAmb(i) = TauAmb
ELSE
  MaxScour(i) = 9999.0
  ShearStress(i) = 9999.0
  AmbFlux(i) = 9999.0
  MaxFlux(i) = 9999.0
  ShearAmb(i) = 9999.0
ENDIF
END DO

```

```

DO i = TowLocPnt+2, RightBankPnt
  va = Tabs(LevelId,i) ! ambient average channel velocity
  dep = WaterDepth(i) ! local depth at centerline of tow
  D50 = D50Size(i) ! SEDIMENT DIAMETER IN M
  VS = D50Vel(i) ! SEDIMENT FALL VELOCITY IN M/SEC
  Y = ABS(Distance(i)) ! distance from centerline for near vessel scour only
  VRMAX = VelChange(i) * direct * (-1.0) ! max return vel at point of interest
  VACell = Tabs(LevelId,i)
  VABOTT = 0.7 * VACell ! bottom velocity- set equal to .7*va
  deltime = .1 ! time step for computations- fix at .1 unless too slow
  IF( D50 .LT. 999.0 ) THEN
    CALL VRSCOUR(VG,VRMAX,VA,DEP,BARLEN,DELTIME,D50,VS,VNU,YLAMB,
*      ETAMAX,iTRANTYPE,TauMax,VACell,CNewMax,Camb,
*      TauAmb)
    MaxScour(i) = ETAMAX
    ShearStress(i) = TauMax
    AmbFlux(i) = CAMB
    MaxFlux(i) = CNEWMAX
    ShearAmb(i) = TauAmb
  ELSE
    MaxScour(i) = 9999.0
    ShearStress(i) = 9999.0
    AmbFlux(i) = 9999.0
    MaxFlux(i) = 9999.0
    ShearAmb(i) = 9999.0
  ENDIF
END DO

```

```

c*****
c////////////////////////////////////^////////////////////////////////////

```

```

c\////////////////////////////////////
c*****

```

```

C ==
C == Initialize Variables and Call Scour Routines ==
C =====

```

```

DO i = LeftBankPnt, RightBankPnt
  IF(VelChange(i) .GE. 10.0) GOTO 855
  IF(DrawDown(i) .GE. 2.0) GOTO 856
  IF(Distance(i) .GE. 10000.0) GOTO 857
  IF(SecWaveHgt(i) .GE. 0.42) GOTO 858
END DO

```

```

c WRITE(8,880) ErrorCode,RiverMile,DirLoc,VelLoc,SizeLoc,
c * DraftLoc,KOLoc,PoolLevel,TowLoc,DirLoc,VelLoc,
c * SizeLoc,DraftLoc,KOLoc,PoolLevel,TowLoc,ATOTAL,
c * ALEFT,BTOTAL,BLEFT,B,L,D,V,VAM,
c * MaxDepth,HSave1,HSave2,U1,ZT

```

```

DO i = LeftBankPnt, RightBankPnt
  WRITE(9, 950) RiverMile, DirLoc, VelLoc, SizeLoc, DraftLoc,
  * KOLoc, PoolLevel, TowLoc, DirLoc, VelLoc,
  * SizeLoc, DraftLoc, KOLoc, PoolLevel, TowLoc,
  * BinLabel(i), WaterDepth(i), VelChange(i),
  * DrawDown(i), Distance(i), SecWaveHgt(i),
  * MaxScour(i), ShearStress(i), AmbFlux(i),
  * MaxFlux(i), ShearAmb(i)
END DO

```

```

950 format(f5.1,'',a1,'',a1,'',a1,'',a1,'',a1,'', a1,'',
  * a1,'',a1,a1,a1,a1,a1,'', a1,a1,'',a10,'',f5.2,'',
  * f6.3,'',f6.3,'',f8.1,'',f6.3,'',f9.4,'',f9.4,'',
  * f11.6,'',f11.6,'',f9.4)
GOTO 99

```

```

862 CONTINUE
  ErrorCode = 'Rbnk' !Could not find Right Bank
  GOTO 850
861 CONTINUE
  ErrorCode = 'Lbnk' !Could not find Left Bank
  GOTO 850
860 CONTINUE
  ErrorCode = 'TLoc' !Could not find Tow Loc
  GOTO 850
859 CONTINUE
  ErrorCode = 'Tran' !Could not find Transect
  GOTO 850
858 CONTINUE

```

```

        ErrorCode = 'SecW' !Secondary Wave Height is to high
        GOTO 850
857 CONTINUE
        ErrorCode = 'Dist' !Distance is to high
        GOTO 850
856 CONTINUE
        ErrorCode = 'DDwn' !DrawDown is to high
        GOTO 850
855 CONTINUE
        ErrorCode = 'RVel' !Return Velocity is to high
        GOTO 850
854 CONTINUE
        ErrorCode = 'TDep' !Water depth at the Tow is too shallow
        GOTO 850
853 CONTINUE
        ErrorCode = 'LP3 ' !Endless Loop Occurred in Loop 3
        GOTO 850
852 CONTINUE
        ErrorCode = 'LP2 ' !Endless Loop Occurred in Loop 2
        GOTO 850
c851 CONTINUE
c   ErrorCode = 'LP1 ' !Endless Loop Occurred in Loop 1
c   GOTO 850
850 CONTINUE
        ErrorCount = ErrorCount + 1
        WRITE(8,880) ErrorCode,RiverMile,DirLoc,VelLoc,SizeLoc,
*         DraftLoc,KOLoc,PoolLevel,TowLoc,DirLoc,VelLoc,
*         SizeLoc,DraftLoc,KOLoc,PoolLevel,TowLoc,ATOTAL,
*         ALEFT,BTOTAL,BLEFT,B,L,D,V,VAM,
*         MaxDepth,HSave1,HSave2,U1,ZT
c   WRITE(8,881) RiverMile
c   WRITE(8,*) ''
c   WRITE(8,882) ATOTAL, ALEFT
c   WRITE(8,883) BTOTAL, BLEFT
c   WRITE(8,*) ''
c   WRITE(8,884) B, D
c   WRITE(8,885) L
c   WRITE(8,*) ''
c   WRITE(8,886) VG
c   WRITE(8,887) VAM
c   WRITE(8,*) ''
c   WRITE(8,888) DirLoc, VelLoc, SizeLoc
c   WRITE(8,889) DraftLoc, KOLoc, PoolLevel
c   WRITE(8,890) TowLoc
c   WRITE(8,*) ''
c   WRITE(8,*) ''
c   WRITE(8,*) ''
880 FORMAT(a4,',','f5.1,',',a1,',',a1,',',a1,',',a1,',',a1,',',a1,
*         ',a1,',',a1,a1,a1,a1,a1,',',a1,a1,',',f9.1,',',f9.1,
*         ',f9.1,',',f9.1,',',f8.2,',',f8.2,',',f8.3,',',f8.3,

```

```

      *      ',f8.3,',f8.3,',f5.2,',f5.2,',f6.3,',f6.3)
881 FORMAT('Transect: ', f5.1)
882 FORMAT('  CHANNEL TOTAL AREA ', f8.1, ' SQ M ',
      *    ' AREA LEFT OF TOW      ', f8.1, ' SQ M ')
883 FORMAT('      TOTAL WIDTH ', f8.1, ' METERS',
      *    ' DISTANCE, LEFT BANK TO TOW ', f8.1, ' METERS')
884 FORMAT(' TOW  WIDTH      ', f8.1, ' METERS',
      *    ' DRAFT          ', f8.1, ' METERS')
885 FORMAT('      LENGTH      ', f8.1, ' METERS')
886 FORMAT(' TOW SPEED RELATIVE TO GROUND ', f8.1, ' M/SEC')
887 FORMAT(' AVERAGE CHANNEL VELOCITY(+=U, -=D)', f8.1, ' M/SEC')
888 FORMAT(' Direction: ', a1, ' Speed: ', a1,
      *    ' Size: ', a1)
889 FORMAT(' Draft: ', a1, ' K/O: ', a1,
      *    ' Stage: ', a1)
890 FORMAT(' Tow Loc: ', a1)

```

```

99 CONTINUE
  END DO
149 CONTINUE

```

```

CLOSE(1)
CLOSE(8)
CLOSE(9)

```

```

C  =====
C  ==          Program Termination Point          ==
C  ==

```

```

STOP ' NORMAL STOP CONDITIONS '
END

```

```

C  =====
C  ==          Program Termination Point          ==
C  =====

```

```

      subroutine vrscour(vg,vrmax,va,dep,barlen,delttime,d50,vs,vnu,
& Ylamb, ETAMAX, iTRANTYPE,TauMax,VACell,CNewMax,Camb, TauAmb)

```

```

*
*  SUBROUTINE VRSCOUR DEFINES THE SCOUR FOR THE ZONE AWAY FROM THE
VESSEL
*
  DIMension vrhis(19, 2)
*
*  DIMENSIONLESS RETURN VELOCITY DISTRIBUTION FROM KAMPSVILLE
REPORT
*

```

```

vrhis(1, 1) = 0
vrhis(1, 2) = 0
vrhis(2, 1) = .2
vrhis(2, 2) = .02
vrhis(3, 1) = .4
vrhis(3, 2) = .1
vrhis(4, 1) = .6
vrhis(4, 2) = .21
vrhis(5, 1) = .7
vrhis(5, 2) = .34
vrhis(6, 1) = .8
vrhis(6, 2) = .5
vrhis(7, 1) = .9
vrhis(7, 2) = .64
vrhis(8, 1) = 1.0
vrhis(8, 2) = .77
vrhis(9, 1) = 1.1
vrhis(9, 2) = .83
vrhis(10, 1) = 1.2
vrhis(10, 2) = .86
vrhis(11, 1) = 1.3
vrhis(11, 2) = .9
vrhis(12, 1) = 1.4
vrhis(12, 2) = .95
vrhis(13, 1) = 1.5
vrhis(13, 2) = 1.0
vrhis(14, 1) = 1.6
vrhis(14, 2) = .92
vrhis(15, 1) = 1.8
vrhis(15, 2) = .65
vrhis(16, 1) = 2
vrhis(16, 2) = .36
vrhis(17, 1) = 2.2
vrhis(17, 2) = .07
vrhis(18, 1) = 2.3
vrhis(18, 2) = .001
vrhis(19, 1) = 50
vrhis(19, 2) = 0
*
* INITIALIZE VARIABLES
*
c  GRAV = 9.805
   TauMax = 0.0
   CNewMax = 0.0
   rho = 1000
   cold = 0
   ETA = 0.0
   ETAMAX = 0.0
   TIMSCOUR = 0
   timbargo = barlen / ABS(vg)

```

```

numtime = INT(2.3 * timbarga / deltime) + 2
cfc = .06 / (LOG10(12.0 * dep / (3.0 * d50))) ** 2
cfr = (2.87 + 1.58 * LOG10(1.0 / (3.0 * d50))) ** (-2.5)

*
* BEGIN ITERATION AT EACH TIME STEP
*
DO 45, I = 1,numtime
timerat = TIMSCOUR / timbarga
vrrat = 0.0
IF (timerat.EQ.0) GOTO 20
*
* THIS LOOP FINDS VRRAT AT EACH TIMERAT
*
DO 10, II = 1,19
IF (timerat.GT.vrhis(II, 1)) GOTO 10
VRAT = (timerat - vrhis(II - 1, 1)) / (vrhis(II,1)-vrhis(II-1,1))
VRRAT = vrhis(II - 1, 2) + VRAT * (vrhis(II, 2) -vrhis(II-1, 2))
GOTO 20
10 CONTINUE
20 CONTINUE
vr = VRRAT * VRMAX
*
* COMPUTE SHEAR USING BLAAUW ET AL (1984)
*
TAU = .5 * rho * cfc * (VaCell + (cfr / cfc) ** .5 * vr) ** 2 !!! max(tau) = shear factor
IF(TAU .GT. TauMax) TauMax = TAU
USTAR = (TAU/RHO)**.5

VAcker = ABS(VaCell + (cfr / cfc) ** .5 * vr)
*
* CALL DESIRED TRANSPORT EQUATION
*
IF(iTRANTYPE.EQ.1)CALL ACKER(USTAR,DEP,D50,VNU,YLAMB,ETA,CNew,
* VAcke)
IF(iTRANTYPE.EQ.2)CALL GARCIA(USTAR,VS,DEP,D50,VNU,YLAMB,ETA,
* COLD,DELTIME,VG,CEQ,TROOLD,ESOLD)

IF(CEQ .GT. CNewMax) CNewMax = CEQ

IF(i .EQ. 1) THEN
Camb = CEQ
TauAmb = Tau
END IF
*
* TEST ETA TO SEE IF EQUAL TO MAX SCOUR
*
IF(ETA.LT.ETAMAX) ETAMAX = ETA
30 TIMSCOUR = TIMSCOUR + deltime
45 CONTINUE

```

```

*
*
*
1000 END

*
  subroutine propscou(ako, dp, vg, va, direct, barbeam, draft,
& barlen,tbl,thrust,dep, vabott, d50,y,ETAMAX,VNU,YLAMB,VS,
& deltime,iTRANTYPE,TauMax,PSPACE,VACell,CNewMax,Camb,TauAmb)
*
* SUBROUTINE PROPSCOU DEFINES SCOUR BENEATH VESSEL
*
  DIMension taup(12, 5), taub(9, 2)
  CHARACTER AKO * 5
*
* DIMENSIONLESS ARRAY FOR BOW SHEAR DISTRIBUTION
*
  taub(1, 1) = 0.0
  taub(1, 2) = -1.17
  taub(2, 1) = .25
  taub(2, 2) = -.73
  taub(3, 1) = .5
  taub(3, 2) = -.51
  taub(4, 1) = .75
  taub(4, 2) = -.33
  taub(5, 1) = 1.0
  taub(5, 2) = 0.0
  taub(6, 1) = .75
  taub(6, 2) = .37
  taub(7, 1) = .5
  taub(7, 2) = .67
  taub(8, 1) = .25
  taub(8, 2) = 1.41
  taub(9, 1) = 0.0
  taub(9, 2) = 3.41
*
* DIMENSIONLESS array for propeller SHEAR DISTRIBUTION
*
  taup(1, 1) = 0.0
  taup(1, 2) = -6.0
  taup(1, 3) = -25.0
  taup(1, 4) = -50.0
  taup(1, 5) = 0.0
  taup(2, 1) = .1
  taup(2, 2) = -3.3
  taup(2, 3) = -15.0
  taup(2, 4) = -30.0
  taup(2, 5) = 0.0
  taup(3, 1) = .25
  taup(3, 2) = -2.1

```

```

taup(3, 3) = -6.4
taup(3, 4) = -11
taup(3, 5) = 0
taup(4, 1) = .5
taup(4, 2) = -1.4
taup(4, 3) = -1.5
taup(4, 4) = -2.7
taup(4, 5) = 0
taup(5, 1) = .75
taup(5, 2) = -.54
taup(5, 3) = -.6
taup(5, 4) = -.7
taup(5, 5) = 0
taup(6, 1) = 1.0
taup(6, 2) = 0
taup(6, 3) = 0
taup(6, 4) = 0
taup(6, 5) = 0
taup(7, 1) = .75
taup(7, 2) = .6
taup(7, 3) = 1.3
taup(7, 4) = 1.7
taup(7, 5) = 0
taup(8, 1) = .5
taup(8, 2) = 2.2
taup(8, 3) = 9.2
taup(8, 4) = 15
taup(8, 5) = 0
taup(9, 1) = .25
taup(9, 2) = 11
taup(9, 3) = 37
taup(9, 4) = 60
taup(9, 5) = 0
taup(10, 1) = .1
taup(10, 2) = 23
taup(10, 3) = 93
taup(10, 4) = 125
taup(10, 5) = 0
taup(11, 1) = .05
taup(11, 2) = 75
taup(11, 3) = 153
taup(11, 4) = 175
taup(11, 5) = 0
taup(12, 1) = 0
taup(12, 2) = 230
taup(12, 3) = 230
taup(12, 4) = 230
taup(12, 5) = 0

```

*

* BEGIN INPUT


```

*
* OPEN(3,"temp.dat", FORM='FORMATTED', STATUS='UNKNOWN')
*
  IF(AKO.EQ.'k') AKO = 'K'
  IF(AKO.EQ.'o') AKO = 'O'
  TauMax = 0.0
  CNewMax = 0.0
  cold = 0
c   vg = vg * DIRECT
c   PSPACE = 6      !!! FIX
c   GRAV = 9.805
  SETBACK = 5
c   THRUST = THRUST / 2
  hp = dep - DP / 2
*
*   TAUFAC IS THE RATIO USED TO ADJUST PHYSICAL MODEL VALUES FROM THE
*   SMOOTH BOUNDARY TO THE ROUGH BOUNDARY FOUND IN THE FIELD AND
TO
*   ACCOUNT FOR THE GREATER TURBULENCE FOUND IN PROPELLER JETS
*   COMPARED TO OPEN CHANNEL FLOW USED TO DEVELOP TRANSPORT
EQUATIONS
*
  taufac = 7.87*(d50)**(.18)
  ETAMAX = 0.0
  ETA=0.0
15 continue
  XSPc = ABS(delttime * vg)
  XBEGIN = 0
  numx = INT((barlen + tbl + .05/D50) / XSPc)
  NUMY = 1
  rho = 999.8

  CFCamb = 0.06/(LOG10(12.0*DEP/(3.0*D50)))**2
  TAUAMB = 0.5*RHO*CFCAMB*VACell**2
  USTAR = (TAUAMB/RHO)**0.5
  VAcker = VACell
  CALL GARCIA(USTAR,VS,DEP,D50,VNU,YLAMB,ETA,
*           COLD,DELTIME,VG,CEQ,TROOLD,ESOLD)
  CAMB = CEQ

*
*   SET KORT OR OPEN PARAMETERS
*
  IF(AKO.EQ.'K') GO TO 12
*
*   THIS SECTION FOR OPEN WHEEL
  D0 = .71 * DP
  E = .43
  CFUNC = .5
  GOTO 17

```

```

12  CONTINUE
*   THIS SECTION FOR KORT
    D0 = DP
    E = .58
    CFUNC = .25
17  CONTINUE
*
*   compute PEAK BOW SHEAR
*
    DOD = dep / DRAFT
    CBOWC = .0118
    CBOWP = -2.85
    IF (DIRECT.GT.0) GOTO 78
    CBOWC = .0148
78  CBOW = CBOWC * DOD** CBOWP
    TAUBOWP = taufac*10000 * CBOW * (va - vg) ** 2.0
*
*   END OF PEAK BOW COMPUTATIONS
*
*   COMPUTE VELOCITY EXITING PROPELLER
*
    U2 = 1.13 / D0 * (THRUST / rho) ** .5
ccc  PRINT *, 'U2 = '
ccc  WRITE(*,*) U2
*
*   COMPUTE WAKE VELOCITY AT PEAK PROP VELOCITY
*
    xprmax = hp / .1
    vwakamax = -1.0 * (va - vg) * (.78) * (DRAFT / dep) ** (1.81)
ccc  PRINT
ccc  PRINT *, 'MAXIMUM WAKE VELOCITY REL TO AMB COND. = '
ccc  WRITE(*,*) VWAKAMAX
ccc  PRINT *, 'MAX WAKE VEL PLUS AMB BOT VEL = '
    TEMP = vwakamax + VABOTT
cc   WRITE(*,*) TEMP
    IF (U2.EQ.0) GOTO 40
*
*   COMPUTE PEAK PROPELLER VELOCITY
*
    f = (1.0 - CFUNC * (ABS(va - vg) / U2) * (hp / DP) ** 1.5)
    GOTO 45
40  f = 0
45  IF(f.LT.0) f = 0
    vpropmax = E * (DP / hp) * U2 * f
*
*   compute wake vel at hp/.1 behind towboat
*
    vwake1 = (1.0 - .0075 * xprmax / DRAFT)
    vwakegx = vwakamax * vwake1 + VABOTT
30  CONTINUE

```

```

ccc  PRINT *, 'WAKE VEL AT HP/X=.1 PLUS AMBIENT BOT VEL = '
ccc  WRITE(*,*) vwakegx
*
*  compute max resultant vel VRES
*
ccc  VRES = -1 * DIRECT * vpropmax + vwakegx
ccc  PRINT *, 'MAX PROPELLER/WAKE/AMB VEL AT HP/X = 0.1 '
ccc  WRITE(*,*) VRES
*
*  VELSHEAR IS THE VELOCITY USED TO DETERMINE THE SHEAR AND DEFINES
*  THE VELOCITY CHANGE FROM THE WAKE WHICH IS GOING ONE DIRECTION
*  AND THE PROPELLER JET WHICH IS GOING THE OPPOSITE DIRECTION
*
Velshear = ABS(vpropmax) + .5 * ABS(vwakegx)
CFPROP = .01 * DP / hp
PROPSH = 0.5 * 10000.0 * CFPROP * Velshear ** 2.0
ccc  PRINT *, 'MAXIMUM PROPELLER SHEAR IN PASCALS = '
ccc  WRITE(*,*) taufac*PROPSH/10  !!! Shear var
ccc  WRITE(*,*) 'SHEAR FACTOR = ',TAUFAC
*
*
*
*  start distribution- y measured from cl of tow, x measured
*  from bow of barges
*
*
36  CONTINUE
35  CONTINUE
*
*  START ITERATION ON X,Y
*
DO 350, I = 1, NUMY
X = XBEGIN
DO 360, J = 1, numx
tau = 0.0
*
*  COMPUTE BOW SHEAR DISTRIBUTION
*
XPEAKBOW = 10
IF (Y.GT.BARBEAM / 2) GOTO 229
XRATBOW = (X - XPEAKBOW) / dep
IF (XRATBOW.LE. - 1.17) GOTO 229
IF (XRATBOW.GT.3.41) GOTO 229
DO 222, JK = 1, 8
IF (XRATBOW.GT.taub(JK + 1, 2)) GOTO 222
TEMP1 = (taub(JK + 1, 1) - taub(JK, 1))
TEMP2 = (XRATBOW - taub(JK, 2)) / (taub(JK + 1, 2) - taub(JK, 2))
TAUBRAT = taub(JK, 1) + TEMP1 * TEMP2
GOTO 228
222  CONTINUE

```

```

228 TAU = TAUBRAT * TAUBOWP
229 CONTINUE
*
*   END BOW SHEAR
*
*   compute wake dist
*
    vwakegx = 0.0
    IF (Y.GT.BARBEAM / 2) GOTO 100
    coef = X - barlen
    IF (coef.LE.0) GOTO 100
    xlim = coef
    IF (xlim.GT.0.1) xlim = .1
    cfc = .06 / (LOG10(12.0 * dep / (3.0 * d50))) ** 2.0
    cfr = (2.87 + 1.58 * LOG10(1.0 / (3.0 * d50))) ** (-2.5)
    IF ((x - barlen).GT.tbl) GO TO 110
    deca = coef / tbl
    vwakegx = vwakamax * deca + VABOTT
    GOTO 120
110  deca = (1.0 + .0075 * (tbl / DRAFT) - .0075 * (X - barlen) / DRAFT)
    IF (deca.LT.0.0) deca = 0.0
    vwakegx = vwakamax * deca + VABOTT
120  ctemp = .5 * 10.0 * CFC * rho
    TAU = ctemp * (va + ((CFR/CFC)**.5) * vwakamax * deca) ** 2.0
*
100  CONTINUE
*   END WAKE DIST
*
*   BEGIN PROP DIST
*
    taurat = 0.0
    IF (x.lt.barlen) GOTO 700
*
*   COMPUTE LATERAL PEAK SHEAR
*
    YDP = Y / DP
    DPHP = DP / hp
    IF (DPHP.gt.1.2) DPHP = 1.2
    IF (DPHP.lt.0.48) DPHP = .48
    IF (YDP.gt.0.547) GOTO 200
*   LINEAR PORTION HERE
    SHRATY0 = 1.207 - .653 * DPHP
    SHRATY = 1.0 - (.547 - YDP) / .547 * (1.0 - SHRATY0)
    peaksh = SHRATY * PROPSH
    GOTO 400
200  IF (YDP.gt.1.095) GOTO 300
*   SHEAR = PEAK SHEAR HERE
    peaksh = PROPSH
    GOTO 400
300  CONTINUE

```

```

* EXPONENTIAL SHEAR HERE
C5 = .0221 * 2.7183 ** (3.14 * DPHP)
SHRATY = 2.7183 ** ((-1.0 * C5) * ((Y - PSPACE / 2) / DP) ** 2)
peaksh = SHRATY * PROPSH
*
* END FINDING LATERAL PEAK SHEAR
*
400 CONTINUE
*
* COMPUTE LONGITUDINAL SHEAR FROM PEAK LATERAL SHEAR
*
XPEAK = barlen + tbl + hp / .2 - SETBACK
XRAT = (X - XPEAK) / hp
*
* PS IS ONLY USED TO SELECT WHICH LONGITUDINAL DISTRIBUTION TO USE
* ALL TAU IN THIS SECTION ARE IN DYNES/SQ CM AND HAVE NOT BEEN
ADJUSTED
* BY TAUFAC. THIS IS NECESSARY TO FIT THE RANGES OF PS FOR FITTING THE
* DISTRIBUTIONS.
PS = peaksh
IF (peaksh.gt.1000.0) PS = 1000.0
IF (peaksh.lt.69.0) PS = 69.0
IF (PS.lt.215.0) GOTO 500
* THIS PART IS FOR PS FROM 1000-215
do 450, jj = 1,12
TEMP = (1000.0 - PS) / 785.0 * (taup(JJ, 3) - taup(JJ, 2))
taup(JJ, 5) = taup(JJ, 2) + TEMP
450 CONTINUE
GOTO 600
500 CONTINUE
* THIS PART FOR PS <215 TO 69
DO 550, kk = 1,12
TEMP = (215.0 - PS) / 146.0 * (taup(KK, 4) - taup(KK, 3))
taup(KK, 5) = taup(KK, 3) + TEMP
550 CONTINUE
600 IF (XRAT.LE.taup(1, 5)) GOTO 900
IF (XRAT.GE.230) GOTO 900
DO 650 k = 1,11
IF (XRAT.GT.taup(k + 1, 5)) GOTO 650
TEMP1 = (taup(k + 1, 1) - taup(k, 1))
TEMP2 = TEMP1 * (XRAT - taup(k, 5)) / (taup(k + 1, 5) - taup(k, 5))
taurat = taup(k, 1) + TEMP2
GOTO 700
650 CONTINUE
*
* END LONGITUDINAL DISTRIBUTION
*
* SET TAU TO MAX OF PROP OR WAKE SHEAR AND ADJUST BY TAUFAC
* (TAU STILL IN DYNES/SQ CM)
*

```

```

700 IF (taufac*taurat * peaksh.GT.TAU) TAU =taufac*
    & taurat * peaksh
900 CONTINUE
    IF((0.1*TAU) .GT. TauMax) TauMax = TAU*0.1

*
*   COMPUTE SHEAR VEL USING TAU IN PASCALS
*
ustar = (0.1*TAU / rho) ** .5
*
IF(iTRANSTYPE.EQ.1)CALL ACKER(USTAR,DEP,D50,VNU,YLAMB,ETA,CNEW,
* VACKER)
IF(iTRANSTYPE.EQ.2)CALL GARCIA(USTAR,VS,DEP,D50,VNU,YLAMB,ETA,
& COLD,DELTIME,VG,CEQ,TROOLD,ESOLD)

IF(CEQ .GT. CNewMax) CNewMax = CEQ

*
*   TEST FOR MAX SCOUR
*
IF(ETA.LT.ETAMAX) ETAMAX = ETA
IF (TauMax .LT. TauAmb) TauMax = TauAmb
*
*   INCREMENT TIME AND X
*
29 TIMSCOUR = TIMSCOUR + deltime
   X = X + XSPc
360 CONTINUE
150 CONTINUE
350 CONTINUE
*
*   END ITERATION ON X,Y
*
1000 END

```

```

SUBROUTINE ACKER(USTAR,DEP,D50,VNU,YLAMB,ETA,cnew,vacker)
GRAV = 9.805
IF (ustar.EQ.0.0) GOTO 25
*
*   compute equivalent vbar based on keulegan equation
*
c   vbar = ustar * 5.75 * LOG10(11.1 * dep / (3 * d50))
*
*   ackers white equation
*
dgr = d50 * ((GRAV * 1.65 / VNU ** 2) ** .3333)
IF(dgr.LE.60.0) GO TO 22
*   COARSE PARAMETERS

```

```

      EN = 0.0
      A = .17
      EM = 1.78
      C = .025
      GOTO 23
22  CONTINUE
*   TRANSITION PARAMETERS
      EN = 1 - .56 * LOG10(dgr)
      A = .23 / (dgr)**.5 + .14
      EM = 6.83 / dgr + 1.67
      Cc = 2.79 * LOG10(dgr) - 0.98*(LOG10(dgr)) ** 2 - 3.46
      C = 10 ** Cc
23  CONTINUE
*   SEDIMENT MOBILITY
      FGR1 = (GRAV * d50 * 1.65)**.5
      FGR11 = ustar ** EN / FGR1
      FGR2 = 10 * dep / d50
      FGR21 = 5.675 * LOG10(FGR2)
      FGR22 = (vacker / FGR21) ** (1 - EN)
      FGR = FGR11 * FGR22
*   TEST FOR ZERO TRANSPORT
      ZQS = FGR / A
      IF(ZQS.LE.1.0) GO TO 25
      GGR = C * (FGR / A - 1) ** EM
*   cnew = SED FLUX IN PARTS/PART by volume
      cnew = (GGR * d50) / (dep * (ustar / vacker) ** EN)
*
*   LIMIT CNEW TO 0.3
*
      if(cnew.gt.0.3) cnew=0.3
      GOTO 26
25  CNEW=0
26  CONTINUE
*
*   EXNER EQ
*
      ETA = -1.0 * (dep * cnew) / (1 - Ylamb)
*
      RETURN
      END
*
      SUBROUTINE GARCIA(USTAR,VS,DEP,D50,VNU,YLAMB,ETA,COLD,DELTIME,VG,
*   CEQ,TROOLD,ESOLD)
*   GARCIA FUNCTION FOR SCOUR
c   WRITE(8,*) VS,COLD,DELTIME,VG
      GRAV = 9.805
      RHO = 1000.0
      IF (ustar.EQ.0.0) GOTO 25
      TAU=RHO*USTAR**2.0
*

```

```

      RP = d50 * (1.65 * GRAV * d50) ** .5 / VNU
*
*   COMPUTE QB WITH ENGELUND AND FREDSOE
*
      taustar = TAU / (rho * GRAV * 1.65 * d50)
      IF(taustar .lt. 0.05) GOTO 25
      Tqstar = 18.74 * (taustar - .05) * (taustar ** .5 - .7 * .05 ** .5)
      Tqb = Tqstar * d50 * (GRAV * 1.65 * d50) ** .5
*
      ZU = (ustar / vs) * RP ** .6
      ACOEF = .00000013
      es = ACOEF * ZU ** 5.0 / (1.0 + (ACOEF / .3) * ZU ** 5.0)
      Tro = 1.0 + 31.5 * (ustar / vs) ** (-1.46)
*
*   TRO LIMITED TO MAX OF GARCIA DATA
*
      IF(TRO.GT.87.7) TRO = 87.7
      CEQ=ES/TRO
      cnew = cold * (1.0 - deltime * VS * TroOLD/ dep) +
&    deltime * VS*esOLD/dep
      IF (cnew .LT. 0.0) cnew = 0.0
      GOTO 26
25  CNEW=0.0
      TQB=0.0
      ZU=0.0
      ES=0.0
      TRO=87.7
      CEQ=0.0
26  CONTINUE
*
*   EXNER EQ
*
      ETA = -1.0 * (Tqb / ABS(vg) + dep * cnew) / (1.0 - Ylamb)
*
      cold = cnew
      esOLD=ES
      troOLD=TRO
      RETURN
      END

```


REPORT DOCUMENTATION PAGEForm Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing this collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number. **PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.**

1. REPORT DATE (DD-MM-YYYY)

September 2002

2. REPORT TYPE

Final report

3. DATES COVERED (From - To)**4. TITLE AND SUBTITLE**

User's Manual for NAVEFF Navigation Effects Model

5a. CONTRACT NUMBER**5b. GRANT NUMBER****5c. PROGRAM ELEMENT NUMBER****6. AUTHOR(S)**

Scott Bourne

5d. PROJECT NUMBER**5e. TASK NUMBER****5f. WORK UNIT NUMBER****7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)**

U.S. Army Engineer Research and Development Center
Environmental Laboratory
3909 Halls Ferry Road
Vicksburg, MS 39180-6199

8. PERFORMING ORGANIZATION REPORT NUMBER

ERDC/EL TR-02-29

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

U.S. Army Engineer District, Rock Island
Rock Island, IL 61204-2004

10. SPONSOR/MONITOR'S ACRONYM(S)**11. SPONSOR/MONITOR'S REPORT NUMBER(S)****12. DISTRIBUTION / AVAILABILITY STATEMENT**

Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES**14. ABSTRACT**

The analysis of system-wide impacts begins with the Navigation Effects (NAVEFF) model. This user guide provides a detailed description of the NAVEFF model installation and execution. The format of the NAVEFF input and output files is documented in this user's guide. The NAVEFF model is a one-dimensional model that estimates the physical forces generated across a cross section caused by shallow draft navigation of a tow boat. NAVEFF is run at each half mile cross section for pools 4, 8, 13, 26, and LaGrange and at 1 mile cross section in the remaining pools. Inputs for NAVEFF are 108 different tow boat combinations, three river stages, and three sailing line locations. Output from NAVEFF includes maximum velocity change, maximum drawdown, maximum wave height, maximum bed scour, and maximum bed shear stress. The output from NAVEFF is one of the primary inputs to the other system ecological models and the sedimentation model.

15. SUBJECT TERMS

NAVEFF Upper Mississippi River System (UMRS)
Navigation Effects Model User Manual

16. SECURITY CLASSIFICATION OF:**a. REPORT**

UNCLASSIFIED

b. ABSTRACT

UNCLASSIFIED

c. THIS PAGE

UNCLASSIFIED

17. LIMITATION OF ABSTRACT**18. NUMBER OF PAGES**

73

19a. NAME OF RESPONSIBLE PERSON**19b. TELEPHONE NUMBER (include area code)**